VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Segmentace hlavy v hloubkových obrazech s využitím grafových metod

Head Segmentation in Depth Images Using Graph-based Methods

2018 Tomáš Bartošek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Tomáš Bartošek**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Segmentace hlavy v hloubkových obrazech s využitím grafových metod
Head Segmentation in Depth Images Using Graph-based Methods

Jazyk vypracování: čeština

Zásady pro vypracování:

V posledních letech se rozvíjí používání 3D senzorů, které vedle obrazové informace poskytují také tzv. hloubkové obrazy uchovávající informaci o vzdálenosti objektů od snímače. Výhodou použití hloubkových dat je nižší citlivost na okolní osvětlení, která je obecně známou nevýhodou při použití klasických RGB obrazů. Cílem diplomové práce je vytvořit software pro segmentaci hlavy člověka v hloubkových obrazech. Segmentace bude realizována pomocí algoritmu založeného na grafových řezech a optimalizovaného pro konkrétní řešení problému segmentace hlavy.

Ve své práci proveďte:
1. Popište zadaný problém a možnosti jeho řešení.
2. Představte segmentační metodu založenou na grafových řezech.
3. Analyzujte řešení a popište potřebnou teorii.
4. Vytvořte aplikaci pro segmentaci hlavy v hloubkových obrazech a proveďte její testování.
5. Zhodnoťte dosažené výsledky.

Seznam doporučené odborné literatury:

[1] Y. Boykov and V. Kolmogorov, *An experimental comparison of min-cut/max- flow algorithms for energy minimization in vision*, in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 9, pp. 1124-1137, Sept. 2004.
[2] G. Slabaugh and G. Unal, *Graph cuts segmentation using an elliptical shape prior*, IEEE International Conference on Image Processing 2005, 2005, pp. II-1222-5.
[3] M. Tang, L. Gorelick, O. Veksler and Y. Boykov, *GrabCut in One Cut*, 2013 IEEE International Conference on Computer Vision, Sydney, VIC, 2013, pp. 1769-1776

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Michael Holuša**

Datum zadání: 01.09.2017

Datum odevzdání: 30.04.2018

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 30. dubna 2018

Barkoš Tomáš

Abstrakt

Cílem této práce je návrh a implementace aplikace, která automatizovaně detekuje a segmentuje lidskou hlavu v hloubkových obrazech s použitím grafových řezů. Výsledný program umožňuje nastavit různé parametry detekce, včetně typů algoritmů; varianty grafových řezů - základní, Grab Cut a One Cut a také Watershed, který používá odlišný způsob segmentace. Detekce mezi navazujícími snímky je optimalizována použitím informací z předchozího obrazu k vyhledání hlavy. Na testovaných snímcích byla s optimální konfigurací dosažena 80% přesnost.

Klíčová slova: Automatická detekce hlavy, segmentace, hloubkový obraz, OpenCV, grafový řez, Grab Cut, One Cut, Watershed

Abstract

The purpose of this thesis is the design and implementation of application, that automatically detects and segments a human head from depth images using graph cuts. Resulting program allows the user to set various parameters of detection, including types of algorithms; variants of graph cuts - basic, Grab Cut and also Watershed which uses different approach to segmentation. Detection between following frames is optimised using the information from previous images to find the head. On tested frames, the accuracy of 80% was achieved using the optimal configuration.

Key Words: Automatic head detection, segmentation, depth image, OpenCV, graph cut, Grab Cut, One Cut, Watershed

Obsah

Seznam použitých zkratek a symbolů **6**

Seznam obrázků **7**

Seznam tabulek **8**

1 Úvod **9**

2 Metody použité při detekci hlavy **10**

2.1 Vstupní hloubkový obraz 12

2.2 Předzpracování obrazu 12

2.3 Detekce objektů 18

2.4 Grafové metody segmentace obrazu 25

2.5 Segmentace obrazu metodou Watershed 32

2.6 Klasifikace objektů 32

2.7 Sledování objektu mezi snímky 38

3 Implementace **43**

3.1 Funkcionalita aplikace 43

3.2 Architektura implementace 45

3.3 Implementace vybraných oblastí 48

4 Výsledky **51**

4.1 Měření 51

4.2 Výsledky kroků detekce hlavy 52

5 Závěr **59**

Literatura **60**

Přílohy **63**

A Souborová příloha **64**

Seznam použitých zkratek a symbolů

OpenCV	–	Open Source Computer Vision, multiplatformní knihovna zaměřující se na zpracování obrazu
3D	–	Trojrozměrný, trojdimenzionální
SI	–	Mezinárodní systém jednotek
FMM	–	Fast marching method, metoda pro řešení okrajových úloh
d.t.	–	Distanční transformace
GMM	–	Gaussian Mixture Model, směs Gaussovských funkcí
BRIEF	–	Binary robust independent elementary features, popis význačných bodů
SURF	–	Speeded up robust features, detekce a popis význačných bodů
ORB	–	Oriented FAST and rotated BRIEF, detekce a popis význačných bodů
kd-strom	–	k-dimenzionální strom
FLANN	–	Fast library for approximate nearest neighbors, softwarová knihovna pro přibližné vyhledávání sousedících bodů
RANSAC	–	Random sample consensus, iterativní metoda pro odhad parametrů nějakého matematického modelu s detekcí odlehlých pozorování
OS	–	Operační Systém
YAML	–	YAML ain't markup language, formát pro serializaci dat v počítači
Pthreads	–	POSIX Threads, implementace podpory paralelizace v počítačích programech podle normy POSIX
POSIX	–	Portable operating system interface, standard rozhraní metod počítačových programů
USB	–	Universal serial bus, standardní sběrnice sloužící pro komunikaci počítače s různými periferiemi
CPU	–	Central Processing Unit, hlavní výpočetní jednotka počítače
RAM	–	Random Access Memory, dočasná paměť počítače
Open CL	–	Open computing language, standard pro paralelní programování počítačů
Cuda	–	Compute unified device architecture, hardwarová a softwarová architektura pro spouštění určitých počítačových programů
Open MP	–	Open multi-processing, rozhraní pro programování paralelních počítačových programů

Seznam obrázků

1 Příklady typů snímků . . . 11
2 Postup detekce hlavy v jednom snímku . . . 11
3 Příklady různých interpretací hodnot pixelů hloubkového obrazu . . . 13
4 Ukázka nežádoucího efektu retušování . . . 14
5 Příklad retušování s omezením rekonstruované plochy . . . 15
6 Příklady příznaků pro kaskádový klasifikátor . . . 18
7 Schéma kaskádového klasifikátoru . . . 19
8 Příklad jednotlivých kroků detekce objektů pomocí distanční transformace . . . 21
9 Příklad výsledku laplaciánu . . . 22
10 Příklad výsledku adaptivního prahu pro detekci objektů . . . 22
11 Distanční transformace aplikována na obrázek 10b . . . 23
12 Příklad vizualizace lokálních maxim distanční transformace . . . 23
13 Příklad průběhu hledání objektů v distanční transformaci . . . 25
14 Reprezentace obrazu grafem . . . 28
15 Příklad výstupu Grab Cut . . . 31
16 Příklad výstupu Watershed . . . 32
17 Model hloubkové kamery a promítání reálného objektu do snímku . . . 37
18 Příklad výstupu úlohy detekce hlavy . . . 44
19 Příklad výstupu úlohy výběr správných výsledků . . . 44
20 Třídní diagram - implementace detekce hlavy v hloubkových obrazech . . . 46
21 Třídní diagram - aplikace . . . 47
22 Třídní diagram - hlavní závislosti úloh aplikace na modulu detekce hlavy . . . 47
23 Ukázka rozdílu v rozlišení barevné hloubky 8 a 16bitového obrazu . . . 52
24 Příklad oddělení objektů . . . 53
25 Měření různých typů segmentace hlavy . . . 55
26 Příklad nepřesnosti Watershed . . . 57
27 Ukázka nedostatečného počtu význačných bodů a párování . . . 58

Seznam tabulek

1 Měření oddělení jednotlivých objektů 53
2 Měření různých filtrů před distanční transformací 54
3 Měření různých způsobů hledání objektů ve výsledku distanční transformace . . . 54
4 Měření různých typů segmentace hlavy 56
5 Měření různých způsobů sledování objektů mezi snímky 57

1 Úvod

V moderní době se prosazuje automatizace různých úkonů pomocí výpočetní techniky. Ve většině úloh jsou nutná vstupní data, která jsou algoritmicky zpracována, aby bylo možno úspěšně dosáhnout daného cíle. Jedním z typů vstupních dat je obraz, jehož zpracování se věnuje samostatná, velmi rozsáhlá disciplína. Fotografie obsahuje velké množství informací, které lidský mozek identifikuje přirozeně, a tak je snaha o extrakci těchto dat pomocí počítačů pochopitelná.

Kromě běžných barevných snímků vznikla zařízení zachycující hloubkový obraz, který obsahuje reliéf prostoru v pohledu kamery. Takový snímek sice vypadá odlišně od běžných fotografií, je ale možno jej zpracovat podobnými metodami, čemuž se věnuje tato práce ve snaze v hloubkovém obraze lokalizovat lidskou hlavu.

Možnosti využití detekce hlavy v hloubkových obrazech pokrývají různé obory lidské činnosti;

- Automobilový průmysl - sledování řidiče, zda se věnuje řízení
- Bezpečnost - střežící systémy.
- Zdravotnictví - sledování pacienta.
- Zábavní průmysl - herní systémy, sledování pozice očí u 3D monitorů využívající paralaxní bariéry.

Potenciální výhody použití tohoto typu snímků spočívají jak v jejich charakteristice, tak fungování zdrojových zařízení; místo barvy hloubkové obrazy popisují topologii prostoru, kterou lze přímo analyzovat. Kamery jsou oproti barevnému typu výrazně méně citlivé na osvětlení; mohou fungovat za tmy, ale ostré světlo dokáže zařízení oslnit.

Tato práce se zaměřuje na plně automatizovanou segmentaci pomocí grafových řezů. Grafové řezy jsou často používanou metodou k segmentaci barevných obrazů s vysokou úspěšností. Cílem práce je vyzkoušení této metody na hloubkových snímcích v několika variantách, které byly původně vyvinuty pro optimalizaci při použití s barevnými obrazy. Nejprve jsou popsány použité metody v navrhovaném přístupu, poté je popsána implementace aplikace a na závěr jsou zhodnoceny dosažené výsledky.

2 Metody použité při detekci hlavy

V této kapitole jsou nejprve popsány jednotlivé kroky automatické detekce a jakým způsobem na sebe navazují. Poté jsou představeny algoritmy jednotlivých úkonů. K některým krokům je uvedeno více metod, mezi kterými je možno v implementované aplikaci přepínat.

Pro větší pochopitelnost textu je vhodné nejprve jednoznačně definovat používané pojmy popisující typy snímků, protože práce se jim hodně věnuje a v některých případech by mohlo dojít ke zmatení čtenáře z důvodu několika možných interpretací.

- Obraz/snímek - dvojrozměrný konečný diskrétní signál o jednom nebo více kanálech. Jeden element signálu je nazýván pixel. Video je interpretováno jako uspořádaná množina snímků, ne jako trojrozměrný signál. Mezi pojmy obraz a snímek se nijak nerozlišuje.

- Světelný obraz - snímek, který zachycuje světelné podmínky ve scéně. Počet kanálů je většinou jeden pro obraz ve stupních šedi (černobílý snímek), tři pro barevné obrázky (nejčastěji červená, zelená a modrá složka) nebo čtyři s přidanou informací o průhlednosti.

- Hloubkový obraz - snímek, jehož hodnoty reprezentují vzdálenost od kamery, má jen jeden kanál - hloubku.

Příklady typů snímků jsou na obrázku 1.

Metody segmentace obrazu pomocí grafových metod[1] nepokrývají samotné nalezení hledaného objektu a ani nijak nepopisují segmentovanou část. Vstupem jsou oblasti popředí - část obrazu, o které je známo, že je součástí hledaného objektu - a pozadí - oblast snímku, o niž je jisté, že se v ní onen objekt nenachází. Z těchto důvodů samotné grafové metody k nalezení lidské hlavy nestačí, ale jsou potřeba kroky předcházející a následující segmentaci.

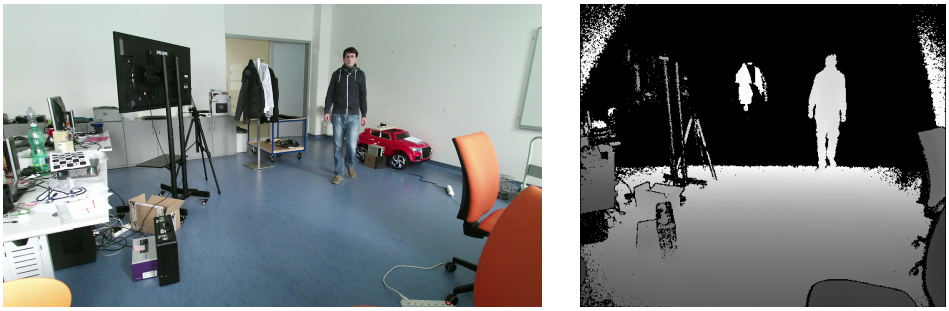
Už na teoretické úrovni je potřeba vzít v potaz, že zdroj hloubkových obrazů není normovaný - je tedy nutno počítat s různými interpretacemi hodnot pixelů snímku. Prakticky je též potřeba uvažovat nedokonalost obrazu, vstupní zařízení může například operovat jen v určitém rozsahu hloubky.

Nalezení hlavy v hloubkových obrazech je tedy vhodné rozdělit do několika po sobě jdoucích kroků, z nichž pouze jeden se věnuje segmentaci pomocí grafových metod:

- Předzpracování obrazu - transformace snímku, aby splňoval předpoklady dalších kroků.

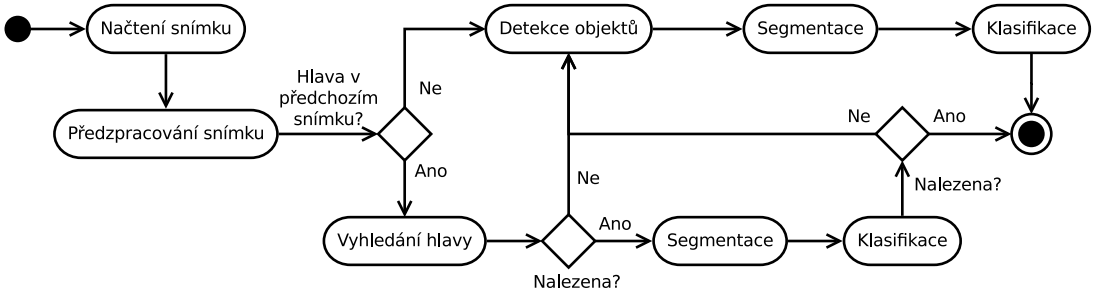
- Detekce objektů - ve snímku jsou nalezeny všechny výrazné oblasti, které potenciálně mohou být hledaným objektem. Výstupem je množina markerů, které nepřesně definují zajímavé oblasti.

- Segmentace - pomocí vstupních markerů jsou jednotlivé objekty přesně segmentovány. Výstupem je kontura pro každý vstupní marker.



(a) Světelný - barevný (b) Hloubkový

Obrázek 1: Příklady typů snímků



Obrázek 2: Postup detekce hlavy v jednom snímku

- Klasifikace - kontury z předchozího kroku a oblasti, které definují, jsou ohodnoceny dle několika kritérií a je vybrána ta, která s největší pravděpodobností reprezentuje lidskou hlavu. Výstupem je jedna nebo žádná kontura v případě, že vypočtené ohodnocení má příliš velkou chybu.

Při použití právě popsaného průběhu detekce hlavy je provedena segmentace pro každý nalezený marker. Segmentaci a klasifikaci je tedy nutno provést tolikrát, kolik je těchto markerů a tím adekvátně roste výpočetní složitost.

Předpokládáme-li, že se poloha hlavy v obraze mezi jednotlivými snímky dramaticky nemění, je možno využít získaných informací z předchozího snímku ke sledování pozice hlavy a tím odstranit nutnost analyzovat všechny ostatní objekty v obraze.

Jsou-li tedy tato data k dispozici, místo detekce objektů se provede algoritmus pro nalezení dostatečně podobné oblasti a po té bude segmentován jen nově nalezený jediný marker. Jestliže tato metoda selže, provede se standardně detekce bez použití dříve získaných informací.

Tento postup je znázorněn diagramem na obrázku 2.

2.1 Vstupní hloubkový obraz

Zdroje hloubkového obrazu mohou být různé; hardwarová kamera, záznam této kamery, syntetický generované snímky a další. V dalším textu se budou používat pojmy „zdroj hloubkového obrazu“, „zařízení pro snímání hloubkových obrazů“ a „kamera“. Vždy je tím myšlen koncept zdroje, jehož výstupem je jeden nebo více hloubkových snímků.

Jak již bylo zmíněno v předchozí kapitole, obraz je dvojrozměrný konečný diskrétní signál, lze jej tedy definovat jako matici I o w_I sloupcích a h_I řádcích s hodnotami z H - oboru hodnot hloubky. V případě hloubkových snímků platí:

- $H = \langle h_{min}, h_{max} \rangle$
- $h_{min}, h_{max} \in R$

Z výše uvedeného vyplývá, že obraz lze ekvivalentně definovat jako funkci $I(x, y) \in H$, kde $I(x, y)$ je definována pro všechna $x \in D_x$ a $y \in D_y$, kde $D_x = \{1, 2, ..., w_I\}$, $D_y = \{1, 2, ..., h_I\}$. Dále se budou používat obě definice I - jako funkce i matice.

Jednotka $h \in H$ je definována zdrojem hloubkového obrazu a obvykle se nejedná o skutečnou, fyzickou, vzdálenost ze soustavy SI.

2.2 Předzpracování obrazu

V pozdější fázi zpracování obrazu mají některé algoritmy jisté předpoklady o vlastnostech vstupního hloubkového snímku. Ne vždy jsou ale tyto podmínky splněny, proto je potřeba zavést předzpracování obrazu, které jej upraví takovým způsobem, aby tyto předpoklady byly dodrženy, nebo se jim alespoň obraz co nejvíce přiblížil.

2.2.1 Interpretace a normalizace hodnot pixelů

Různé zdroje hloubkových obrazů používají odlišnou interpretaci hodnot hloubky; v některých případech jsou pixely s hodnotou v okolí h_{min} použity pro blízké body, zatímco u jiných kamer jde o body nejvzdálenější.

Dále, ne vždy dokáží tato zařízení změřit hloubku. V takovém případě do pixelu uloží nějakou hodnotu $h_u \in H$ definovanou pro indikaci tohoto případu. Příklady různých interpretací jsou na obrázku 3.

Neexistuje jednotná norma, která by určovala hodnoty h_{min}, h_{max}, jejich interpretaci vzhledem ke vzdálenosti od kamery a h_u, byť ve většině případů[1] $h_u = h_{min} = 0.0$. Z těchto důvodů je vhodné stanovit definici normalizovaného snímku pro účely této práce. Ta byla zvolena následovně:

- $h_{min} = 0.0$, body v okolí h_{min} jsou nejvzdálenější.

[1] Nejedná se o závěr formálního výzkumu, ale jednak osobní zkušenost se zařízením a jednak toto naznačují výsledky hledání hloubkových obrazů na internetu.

(a) Originální snímek, h_{max} reprezentuje nejvzdálenější body, $h_u = h_{min}$.

(b) h_{min} pro nejbližší body, h_{max} pro nejvzdálenější

(c) h_{min} pro nejvzdálenější body, h_{max} pro nejbližší

Obrázek 3: Příklady různých interpretací hodnot pixelů hloubkového obrazu

- $h_{max} = 1.0$, body v okolí h_{max} jsou nejbližší.
- $h_u = 0.0$

Protože přesnost algoritmů není ovlivněna volbou interpretace h_{min} a h_{max}, zvolená varianta byla vybrána proto, že vizualizace, kdy h_{min} se zobrazí jako černá barva a h_{max} jako bílá, je pro lidský mozek (subjektivně) snáze představitelnější.

V dalším textu se předpokládá, není-li uvedeno jinak, že obraz je normalizován dle výše uvedené definice. Mělo by být zřejmé, že $h \in H$ je v případě normalizovaného snímku bezrozměrná hodnota.

2.2.2 Nepřečtené hodnoty

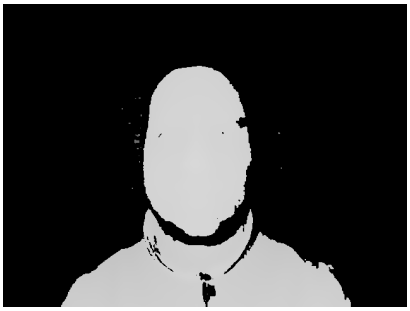
V předchozí části byla zavedena hodnota h_u pro pixely, jejichž vzdálenost se nepodařilo změřit. Tyto body narušují topologickou informaci a jestliže by měly být použity bez speciálního ošetření, algoritmy mohou snadno selhat, protože budou interpretovat tvar objektu chybně.

Protože algoritmy dalších kroků předpokládají, že se hodnota hloubky hledaného objektu nenachází v nejbližším okolí h_{min} a h_{max}, a tedy v okolí h_u, není nutné speciálně detekovat, zda pixel o hodnotě h_u je skutečně nedetekovaný bod, nebo se daný objekt nachází na hranici rozlišovací schopnosti kamery.

Dále je potřeba určit, jak se s těmito body bude zacházet. Prakticky se nabízí dvě možnosti;

- Uložit informaci o těchto pixelech např. ve formě bitové matice a následující algoritmy budou brát tuto skutečnost v potaz.
- Rekonstruovat obraz retušováním.

V ideálním případě je vhodná první varianta, protože rekonstrukce může zanést nepřesnost. Toto by ale vyžadovalo úpravu velkého množství dílčích algoritmů, což je mimo rozsah této práce. Proto je zvolený postup kombinovaný; obraz bude rekonstruován, ale zároveň další kroky budou



(a) Originální snímek

(b) Nevhodně retušovaný výsledek (rekonstrukce pozadí)

Obrázek 4: Ukázka nežádoucího efektu retušování

mít k dispozici informaci o nedetekovaných bodech. Algoritmu retušování se věnuje následující sekce.

Při výběru oblastí pro rekonstrukci je ale potřeba rozlišovat dva případy:

- Bod se nachází mimo rozsah vzdáleností, které je zařízení schopno měřit, tedy je příliš blízko nebo naopak příliš daleko.
- Bod se sice nachází v tomto rozsahu, ale zařízení přesto nebylo schopno určit jeho vzdálenost.

Je žádoucí opravovat pouze body v rozsahu rozlišovací schopnosti zařízení, protože pro korektní rekonstrukci bodů mimo tento rozsah není ve snímku potřebná informace. V takovém případě by objekty hraničící s takovou oblastí ztratily svůj tvar. Příklad tohoto nežádoucího jevu je na obrázku 4.

K identifikaci konkrétního případu ve snímku ale není žádná informace; v obou případech je hodnota pixelů v obraze stejná, h_u. Navrhované řešení je retušovat jen oblasti s hodnotou h_u, jejichž plocha je relativně malá a tedy se s určitou pravděpodobností předpokládá, že se nejedná o pozadí, nýbrž o šum. Obrázek 5 demonstruje výsledek tohoto omezení.

2.2.2.1 Retušování Různé algoritmy retušování jsou popsány v [2], [3] a [4]. [4] navrhuje metodu optimalizovanou pro rychlost výpočtu s výsledky srovnatelnými s komplexnějšími a náročnějšími metodami. Oprava pixelu v bodě p je definována jako

$$I_p = \frac{\sum_{q \in B_\varepsilon(p)} w(p,q)(I_q + \nabla I_q(p-q))}{\sum_{q \in B_\varepsilon(p)}} \tag{1}$$

kde $B_\varepsilon(p)$ je množina známých bodů obrazu I v ε-okolí bodu p, ∇I je gradient obrazu I a $w(p,q)$ je váha hodnoty I_q pro opravu I_p, která je definována jako

$$w(p,q) = dir(p,q)\, dst(p,q)\, lev(p,q) \tag{2}$$

(a) Originální snímek

(b) Maska pro rekonstrukci (růžová - pozadí, zelená - zvolené oblasti pro retušování, modrá - body se změřenými hloubkovými daty.)

(c) Výsledek

Obrázek 5: Příklad retušování s omezením rekonstruované plochy

$$dir(p,q) = \frac{p-q}{|p-q|} N_p \tag{3}$$

$$dst(p,q) = \frac{d_0^2}{|p-q|^2} \tag{4}$$

$$lev(p,q) = \frac{T_0}{1 + |T_p - T_q|} \tag{5}$$

kde T je matice vzdálenosti bodů k nejbližšímu známému, neretušovanému bodu, $N = \nabla T$ je gradient T a T_0, d_0 jsou referenční vzdálenosti, obvykle definované jako konstanta 1 (vzdálenost dvou pixelů). *dir* prosazuje body ve směru N_p, normálového vektoru hranice opravované plochy, *dst* snižuje vliv vzdálenějších bodů a *lev* zvětšuje váhu pixelům v podobné vzdálenosti od kraje opravované plochy.

Algoritmus používá Fast marching method (FMM) pro výpočet T a samotné retušování. Prvním krokem je inicializace používaných hodnot:

- R - množina bodů k retušování.
- Z - množina známých bodů (z P_I), které sousedí s R (čtyřsměrná sousednost). Tato množina reprezentuje body, jejichž okolí je potřeba zpracovat.
- f - matice, která jednotlivým bodům $p \in P_I$ přiřazuje označení:

$$f_p = \begin{cases} F_B & p \in Z \text{ - všechny body hraničící s retušovaou oblastí.} \\ F_K & p \in P_I \setminus (R \cup Z) \text{ - všechny známé body (neretušované), které ale nejsou v } Z. \\ F_U & p \in R \text{ - všechny body k retušování.} \end{cases}$$

- $T_p = \begin{cases} 0 & p \in P_I \setminus R \\ T_{max} & p \in R \end{cases}$

  kde T_{max} je velké číslo, obvykle definované jako $T_{max} = 10^6$.

Množina Z reprezentuje body, jejichž okolí je potřeba zpracovat a zároveň jsou tyto body v f označeny jako F_B. Body ze Z jsou při zpracování přeznačeny na F_K, zatímco jejich okolní body označené jako F_U jsou retušovány, přiřazeny do Z, jejich označení je změněno na F_B a jsou aktualizovány adekvátní hodnoty v T. Algoritmem FMM s retušováním je tedy cyklus:

- Z množiny Z je vyjmut bod p, který má ze všech bodů v Z nejmenší hodnotu T_p.
- Nastavit $f_p = F_K$.
- Pro všechny body q ve čtyřsměrném okolí bodu p u kterých platí, že $f_q \neq F_K$:
  - Pokud $f_p = F_U$, pak nastavit $f_p = F_B$ a retušovat bod I_p dle vzorce (1).
  - Upravit T_q dle vzorce (6); popsáno dále.
  - Přidat q do Z.
- Opakovat, dokud Z není prázdná.

Výpočet T_p, kde $p = (x, y)$, je definován jako

$$\begin{aligned} T_p = T_{(x,y)} = \min(&ee(x-1, y, x, y-1), \\ &ee(x+1, y, x, y-1), \\ &ee(x-1, y, x, y+1), \\ &ee(x+1, y, x, y+1)) \end{aligned} \tag{6}$$

$$ee(x_p, y_p, x_q, y_q) = ee(p, q) = \begin{cases} T_{max} & f_p \neq F_K \wedge f_q \neq F_K \\ T_p + 1 & f_p = F_K \wedge f_q \neq F_K \\ T_q + 1 & f_p \neq F_K \wedge f_q = F_K \\ eek(p, q) & f_p = F_K \wedge f_q = F_K \end{cases} \tag{7}$$

$$eek(p, q) = \begin{cases} s_1 & s_1 \geq T_p \wedge s_1 \geq T_q \\ s_2 & s_2 \geq T_p \wedge s_2 \geq T_q \\ T_{max} & \text{jinak} \end{cases} \tag{8}$$

$$s_1 = 0.5\,(T_p + T_q - r) \tag{9}$$

$$s_2 = s_1 + r \tag{10}$$

$$r = \sqrt{2 - (T_p - T_q)^2} \tag{11}$$

Volba ε, tedy poloměr okolí pro retušování bodu, je závislá na charakteristice retušovaných oblastí, především tloušťce. S větším ε je výsledek více rozmazaný, ale lépe zvládá rozsáhlejší oblasti k opravě. Obvykle se volí mezi třemi až deseti pixely.

2.2.3 Filtrování

Zařízení pro snímání hloubkových obrazů mohou dávat výsledky o různé kvalitě, může být tedy žádoucí obraz dále filtrovat. Tyto filtry mohou být:

- Gaussovské rozostření[5][6] - hodnota pixelu nového obrazu je vážený průměr okolních hodnot pixelů, kde váha vychází z gaussovské funkce:

$$e^{\frac{-(x-\mu_x)^2}{2\sigma_x^2} + \frac{-(y-\mu_y)^2}{2\sigma_y^2}} \tag{12}$$

  kde σ_x a σ_y definují rozptyl, tedy míru rozostření, v osách x a y.

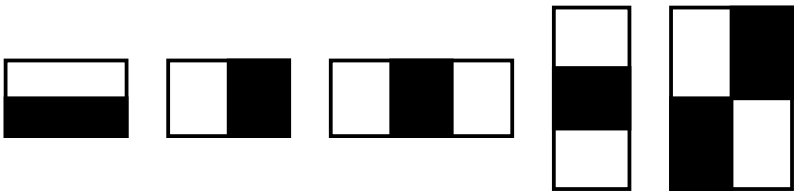
- Rozostření mediánem (median blur)[5] - hodnota pixelu nového obrazu je medián definovaného okolí.

- Non-local means denoising (snížení šumu nelokální střední hodnotou[2])[7] - hodnota nového pixelu je průměr pixelů z podobných oblastí, které mohou být vyhledány mimo blízké okolí původního pixelu.

Gaussovské rozostření je nejčastější způsob, jak snížit šum v barevných obrazech, ale - podobně jako aritmetický průměr - je náchylný na extrémní hodnoty / odlehlá pozorování. Je vhodný pro odstraňování šumu, který má normální rozložení, tedy pokud se hodnota měřených pixelů příliš neodchyluje od skutečné hloubky. Rozostření mediánem je víc robustní v těchto případech a lépe zachovává ostrost hran. V případě, že oblasti s hodnotou h_u jsou obzvlášť malé, může tento filtr nahradit retušování, které pak lze úplně vynechat. Non-local means denoising odstraňuje šum na základě vyhledávání podobných oblastí v obraze.

Výběr konkrétního filtru, nebo zda vůbec bude nějaký použit, záleží na charakteristice výstupu použitého zařízení pro zachytávání hloubkových snímků.

[2] Nebyl nalezen oficiální překlad názvu tohoto algoritmu, proto je použita původní, anglická, verze s vlastním překladem.



Obrázek 6: Příklady příznaků pro kaskádový klasifikátor

2.3 Detekce objektů

Úkolem tohoto kroku je nalézt potenciálně zajímavé objekty, které budou dále podrobeny detailnější segmentaci a klasifikaci.

- Vstup: předzpracovaný hloubkový obraz I.
- Výstup: množina markerů M reprezentující nalezené objekty;
  - $M = \{M_1, M_2, ..., M_n\}$
  - M_i - jeden marker, který je definován jako množina souřadnic m_{ij} pixelů snímku I, které spolu sousedí.
  - n - počet markerů
  - n_i - počet bodů v markeru M_i

Protože primárním cílem je nalezení samotných objektů a výsledky budou dále zpracovány, není v tomto kroku kladen důraz na přesnost a předpokládá se, že M_i nepokrývá celý objekt a nevypovídá o jeho tvaru.

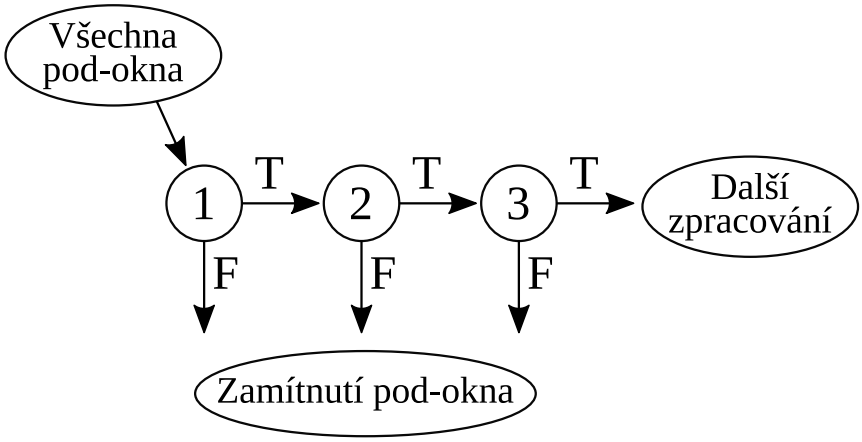
2.3.1 Kaskádový klasifikátor s příznaky Haar

Tento klasifikátor, poprvé popsán v [8], byl vyvinut pro detekci lidské tváře v barevných snímcích a tento úkon zvládá z vysokou přesností. Později byl použit k detekci i jiných typů objektů se střídavou úspěšností.

Klasifikátor pracuje s množinou jednoduchých příznaků[3] (viz obr. 6). Ke každému příznaku je přiřazen práh, který je použit při jeho vyhodnocení - od součtu hodnot pixelů v černé části příznaku je odečten součet hodnot pixelů v bílé oblasti. Je-li výsledek menší než práh, je příznak vyhodnocen kladně.

Klasifikátor je rozdělen do několika vrstev, které se skládají z disjunktních podmnožin příznaků, jež jsou rozděleny tak, aby první vrstvy s co největší šancí zamítly obraz bez tváře. Díky tomu jsou nezajímavé oblasti obrazu rychle zahozeny a teprve slibnější místa jsou podrobena detailnějšímu testu dalších vrstev.

[3] Tyto příznaky se v literatuře občas označují jako „Haarovy příznaky“. Toto pojmenování je ovšem nepřesné, protože Haarova vlnka odpovídá pouze druhému příznaku na obr. 6 a ostatní jsou z něj odvozeny.



Obrázek 7: Schéma kaskádového klasifikátoru

Celý tento algoritmus je aplikován na pod-okna segmentovaného snímku (viz obr. 7). Díky využití tzv. integrálního obrazu[4] má pak vyhodnocení jednoho příznaku konstantní a velmi malou výpočetní složitost. S využitím této techniky spolu s výše popsaným systémem vrstev je detekce i přes velký počet pod-oken velmi rychlá.

Tento algoritmus zároveň řeší hned dva úkoly - nejen detekci objektu, ale rovnou i jeho klasifikaci. Původně byl vyvinut k detekci tváře ve světelných obrazech [8] a natrénovaný klasifikátor k tomuto účelu je volně k dispozici [9]. Byly provedeny experimenty pro použití na klasifikaci jiných typů objektů s různými výsledky.

Aby bylo možno tento detektor využít na hloubková data, je potřeba připravit dostatečně velkou a variabilní množinu trénovacích dat, což je vzhledem k malé dostupnosti hloubkových obrazů mimo rozsah této práce - je potřeba vzít v potaz nejen různé úhly pohledu, ale také mnoho odlišných typů obličejů.

Přesto je možno jej použít v několika situacích:

- K hloubkovému obrazu je k dispozici i adekvátní barevný snímek. Jeho využití sice částečně ruší výhody použití hloubkových obrazů, nicméně je možno jej využít jako doplněk pro detekci v případě vhodných světelných podmínek a po té sledovat hlavu mezi dalšími snímky (popsáno dále).

- Pro měření přesnosti algoritmů používajících jen hloubkový obraz a porovnání výsledku s detekcí v barevném snímku pomocí kaskádového klasifikátoru.

V implementované aplikaci je použit jako alternativní zdroj detekovaných objektů, který je možno volitelně zapnout, je-li zdroj barevných snímků k dispozici.

[4] Pro všechny pozice ve snímku je spočítán součet hodnot všech pixelů v regionu vlevo nahoře nad daným pixelem. Pro výpočet součtu hodnot pixelů v libovolném pod-okně jsou pak potřeba jen čtyři předpřipravené součty v rozích tohoto pod-okna.

2.3.2 Detekce objektů pomocí distanční transformace

Hledání výrazných objektů spočívá v detekci oblastí, které mají podobnou hodnotu pixelů; v případě světelných obrazů se jedná o podobnost jasu, v případě hloubkových snímků se pak hledá podobná hloubka.

Často se detekce objektů skládá z několika kroků, které nejprve připravují snímek na distanční transformaci a její výsledek se dále zpracovává ([10], [11]):

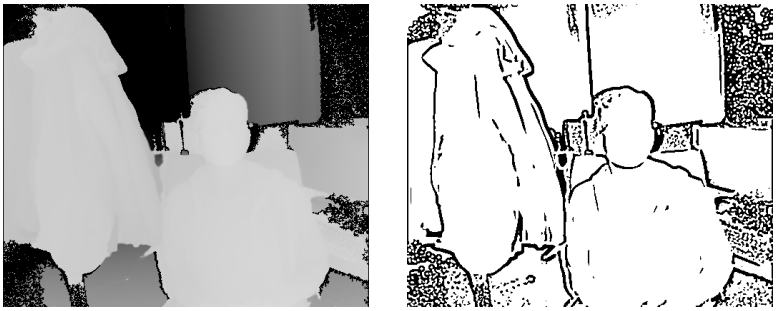
- Příprava snímku, která odděluje jednotlivé objekty - prahování, detekce hran, příp. specifické úpravy, jako je odstranění známého pozadí.
- Distanční transformace.
- Vyhledání oblastí s vyššími hodnotami, které jsou oddělené hodnotami nižšími, např. prahováním.

Vstupem distanční transformace je binární obraz, tj. takový, který nabývá jen dvou hodnot - 0 a 1. Výsledkem je snímek, jehož hodnoty pixelů odpovídají vzdálenosti od dané pozice k nejbližšímu pixelu s nulovou hodnotou v původním obraze. Vyhledáním pixelů s vyššími hodnotami ve výsledku tohoto algoritmu je možno identifikovat rozsáhlejší oblasti s hodnotou 1 v původním binárním obraze. Hlavní výhodou je, že tyto části jsou nalezeny a odděleny i v případě, že oblast není přesně ohraničena a je tedy spojena s jinou částí obrazu.

Při detekci objektů se tato transformace aplikuje na snímek, který jen hrubě klasifikuje pixely zpracovávaného obrazu na popředí a pozadí. Nevýhodou distanční transformace je, že je velmi citlivá na šum; jediný pixel uprostřed oblasti odpovídající nějakému objektu způsobí, že maximální hodnota této oblasti bude výrazně nižší.

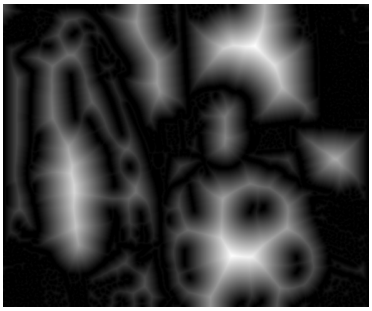
Detekci objektů lze tedy rozdělit do následujících kroků (příklad na obrázku 8):

- Oddělení jednotlivých objektů
  - Vstup: hloubkový obraz
  - Výstup: binární snímek, hodnota 0 odděluje objekty definované hodnotou 1
  - Popis: hloubkový obraz je rozdělen na oblasti, které odpovídají jednotlivým objektům. Objekty nejsou odděleny dokonale a mohou být tedy částečně propojeny. Kvůli vysoké citlivosti distanční transformace na šum je lepší v nejasných případech klasifikovat daný pixel jako objekt. Rozebráno v následující podkapitole.
- Distanční transformace
  - Vstup: binární snímek s hrubým rozdělením obrazu na oblasti jednotlivých objektů
  - Výstup: obraz popisující vzdálenost pixelů od nejbližší pozice oddělující oblasti
- Detekce objektů v distanční transformaci

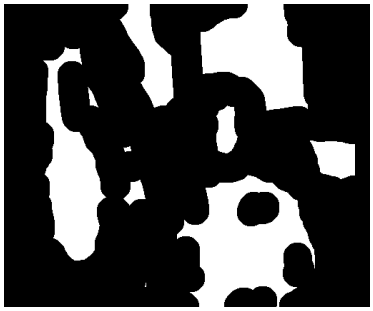


(a) Originální snímek

(b) Oddělení jednotlivých objektů



(c) Distanční transformace



(d) Nalezené objekty pro další zpracování

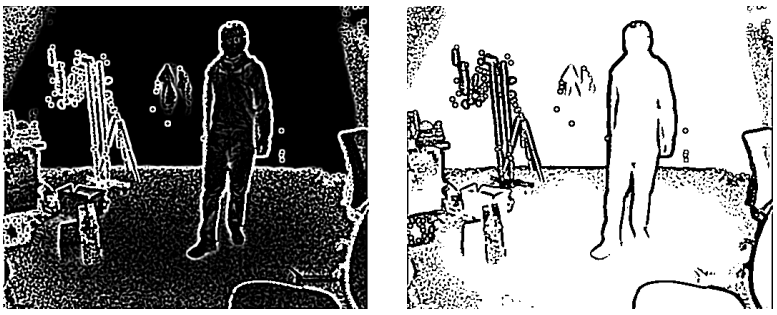
Obrázek 8: Příklad jednotlivých kroků detekce objektů pomocí distanční transformace

- Vstup: výsledek distanční transformace
- Výstup: množina markerů
- Popis: vyhledání oblastí s vyššími hodnotami ve výsledku distanční transformace. Rozebráno v následující podkapitole.

2.3.2.1 Oddělení jednotlivých objektů U světelných obrazů se často používá prahování nebo detektor hran ([10], [11]), což vychází z předpokladu, že hledaný objekt má ve snímku odlišnou intenzitu, než jeho okolí.

V případě hloubkových obrazů je toto splněno za předpokladu, že se celý objekt nachází v podobné vzdálenosti od kamery a pokud je kolem něj volný prostor. V případě hlavy jsou tyto předpoklady většinou platné.

Laplaceův operátor, zkráceně laplacián, je jeden z detektorů hran, který je možno pro tento účel použít [12]. Tato transformace počítá druhou derivaci vstupního signálu, v případě diskrétního obrazu se jedná o aproximaci této derivace. Výsledek laplaciánu lze interpretovat tak, že vysoké kladné hodnoty reprezentují hranu mezi objekty. K tomu, aby byl tento obraz použitelný pro distanční transformaci, je potřeba jej prahovat a invertovat. Protože je Laplaceův operátor citlivý na šum, stejně jako distanční transformace, je nutné aplikovat filtr, například rozostření mediánem. Příklad je na obrázku 9.



(a) Nezpracovaný obraz (b) Po aplikování filtru a inverzního prahu

Obrázek 9: Příklad výsledku laplaciánu

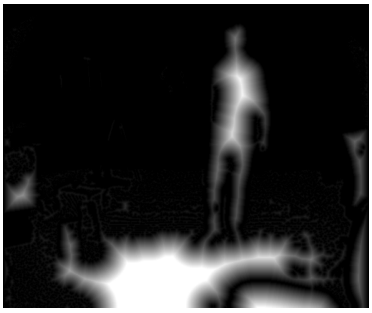
(a) Nezpracovaný obraz (b) Po aplikování filtru

Obrázek 10: Příklad výsledku adaptivního prahu pro detekci objektů

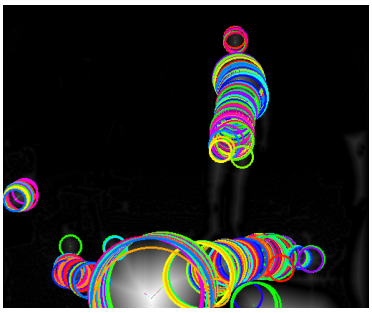
Odlišný přístup k oddělení objektů umožňuje adaptivní prahování [13]. Běžné prahování používá fixní hodnotu prahu; je-li hodnota pixelu větší, na výstup je uložena 1, jinak 0. Adaptivní prahování tuto metodu rozvíjí tak, že prahová hodnota je dynamicky vybrána na základě analýzy okolí pixelu. Jestliže je splněna podmínka, že okolo hledaného objektu je volný prostor, potom je jisté, že hodnoty pixelů reprezentující objekt jsou vyšší, než pixely v nejbližším okolí[5]. Ačkoliv tato metoda není tolik citlivá na šum jako laplacián, i v tomto případě je vhodné aplikovat filtr, než bude obraz podroben distanční transformaci. Jinak výsledek adaptivního prahování není potřeba dále upravovat, snímek je již binární a jeho nulové pixely oddělují objekty. Příklad je na obrázku 10.

2.3.2.2 Prahování distanční transformace Posledním krokem v detekci objektů s použitím distanční transformace je jejich nalezení ve výsledku tohoto algoritmu. Nejjednodušším způsobem je základní prahování; fixní prahová hodnota definuje minimální vzdálenost od středu objektu k nejbližšímu pixelu ohraničující jednotlivé oblasti.

[5] Kdyby byla zvolena obrácená interpretace hodnot pixelů, tj. h_{min} reprezentuje blízké body a h_{max} vzdálené, platilo by toto obráceně a bylo by potřeba výsledek invertovat.



Obrázek 11: Distanční transformace aplikována na obrázek 10b. Obrázek byl dodatečně upraven, aby výsledek transformace byl výraznější.



Obrázek 12: Příklad vizualizace všech výraznějších lokálních maxim distanční transformace. Každá kružnice reprezentuje jedno maximum, poloměr pak vzdálenost od daného bodu k nejbližšímu hraničnímu bodu.

Tato metoda funguje dostatečně dobře, jestliže oddělení objektů - první krok detekce objektů - dokáže nalézt hraniční pixely se všemi okolními objekty. Příklad tohoto případu je na obrázku 8.

2.3.2.3 Analýza vrcholů distanční transformace

Je zjevné, že lidské tělo je z pohledu hloubkových obrazů jedno těleso, a tedy detekce objektů pomocí prahování distanční transformace může selhat, když bude mezi hlavou a trupem rozdíl hloubky příliš malý a nebude tedy mezi nimi nalezena hrana, jak je možno vidět na obrázku 11. Pokud by parametry byly upraveny tak, aby byly detekovány takové slabší hrany, objevil by se i výraznější šum a hrany na obličeji, například v okolí nosu, čímž by se výsledek stal nepoužitelným kvůli citlivosti na šum distanční transformace. Proto je navržen odlišný způsob detekce potenciálních objektů, než je prahování. Tento způsob vychází především z pozorování - a je tedy založen na předpokladu - že oblast hlavy je v distanční transformaci reprezentována lokálním maximem. Příklad tohoto jevu je na obrázku 12.

Každé lokální maximum je definováno svou pozicí ve snímku a hodnotou, která odpovídá vzdálenosti k nejbližšímu bodu pozadí. Lokální maximum tedy lze reprezentovat kružnicí o poloměru rovnému této hodně.

Hlavní myšlenkou je sloučení všech kružnic, které reprezentují jeden objekt, a použít výslednou množinu pro výstupní markery; marker je v tomto případě definován jako množina všech bodů snímku uvnitř kružnice. Navržený algoritmus se skládá ze dvou iterací; v první jsou slučovány body v bezprostřední blízkosti pro rychlou redukci jejich počtu, ve druhé iteraci se pak bere v potaz míra průniku kružnic.

Algoritmus je tedy:

1. Vyhledání všech lokálních maxim, jejichž hodnota je větší než práh r_{min}.

2. Identifikace shluků s použitím metody nejbližšího souseda - bod patří do shluku, jestliže vzdálenost od nejbližšího bodu shluku je menší, než stanovené minimum d_c.

3. Vytvoření redukované množiny lokálních maxim výběrem kružnice s největším poloměrem z každého shluku.

4. Druhá iterace identifikace shluků z redukované množiny lokálních maxim, opět s použitím metody nejbližšího souseda, podmínka sloučení je ale jiná: musí být splněno

$$d < (r_1 + r_2)\, d_{rp} \tag{13}$$

   kde r_1 a r_2 jsou poloměry kružnic, d je jejich vzdálenost a $d_{rp} \in (0.0, 1.0)$ je jejich maximální relativní vzdálenost.

   Zároveň budou odstraněny kružnice, které by měly být sloučeny, ale jsou příliš malé. Podmínka odstranění je

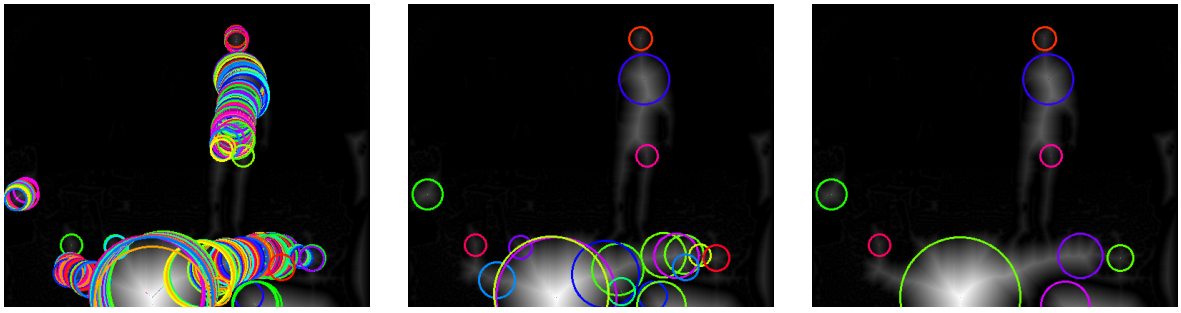
$$r_i < max(r_1, r_2)\, r_d \tag{14}$$

   kde $r_d \in (0.0, 1.0)$ limitní poměr poloměrů pro odstranění a r_i je kružnice z r_1, r_2, která má menší poloměr. Tato část slouží k oddělení shluků kružnic, které sice splňují podmínky sloučení, ale mezi nimi se nachází výrazné zúžení. Detekce těchto zúžení nahrazuje detekci hran, které jsou příliš slabé.

5. Vytvoření výstupní množiny kružnic ze shluků stejným způsobem, jako na konci první iterace.

Algoritmus má tedy následující konfigurační parametry:

- r_{min} - minimální hodnota lokálního maxima (poloměr kružnice), aby vůbec byl tento bod uvažován.
- d_c - maximální vzdálenost kružnic pro první iteraci shlukování.
- d_{rp} - maximální relativní vzdálenost kružnic (vzhledem k součtu jejich poloměrů) pro druhou iteraci shlukování.



(a) Všechna výrazná lokální maxima (b) První iterace shlukování (c) Druhá iterace shlukování

Obrázek 13: Příklad průběhu hledání objektů v distanční transformaci

- r_d - limitní poměr poloměrů pro odstranění kružnic při shlukování ve druhé iteraci.

Příklad průběhu této metody na obrázku 13.

2.3.3 Shrnutí detekce objektů

Byly popsány dva způsoby rychlého a hrubého nalezení potenciálně zajímavých objektů. Prvním z nich je kaskádový klasifikátor Haar, druhý využívá distanční transformaci a skládá se z následujících kroků:

- Oddělení jednotlivých objektů
  - Detekce hran, např. Laplaceovým operátorem
  - Adaptivní prahování
- Distanční transformace
- Detekce objektů ve výsledku distanční transformace
  - Prahování
  - Analýza vrcholů d.t.

2.4 Grafové metody segmentace obrazu

Problém segmentace obrazu je možno transformovat do grafu, ten zpracovat metodami vycházející z teorie grafů a výsledek zpětně interpretovat v doméně obrazu. K tomuto účelu se využívá minimální řez/maximální tok (min-cut/max-flow), popsáno v [14, 15]. [1] popisuje různé algoritmy řešící tento problém, navrhuje algoritmus nový a porovnává jejich výsledky a rychlost.

2.4.1 Základní segmentace grafovým řezem

2.4.1.1 Energetická funkce Grafové řezy (graph cuts) se používají k řešení minimalizace energie

$$E(L) = A(L) + B(L) = \sum_{p \in P_I} A_p(L_p) + \sum_{(p,q) \in N_I} B_{p,q}(L_p, L_q) \tag{15}$$

kde $P_I = \{(x, y) : x \in D_x, y \in D_y\}$ je množina všech souřadnic pixelů snímku I, $L = \{L_p : p \in P_I\}$ definuje segmentaci tak, že $L_p \in \{0, 1\}$ přiřazuje každému pixelu označení 1 - popředí (objekt) nebo 0 - pozadí, N_I je množina dvojic souřadnic pixelů, které spolu sousedí, A_p vrací penalizaci při přiřazení označení L_p pixelu p a $B_{p,q}$ vrací penalizaci přiřazení označení L_p a L_q pixelům p a q. Hledáme tedy takovou segmentaci L, aby byla celková energie minimální.

Chyba A_p vyjadřuje (inverzně) příslušnost pixelu p určitému označení. Zdrojem může být uživatelský vstup $C = \{c_p : p \in P_I, c_p \in R\}$, kde c_p vyjadřuje míru toho, zda se jedná o popředí ($c_p > 0$) nebo pozadí ($c_p \leq 0$)[6]. C lze využít přímo k definici A_p:

$$A_p(L_p) = \begin{cases} max(c_p, 0), & L_p = 0 \\ max(-c_p, 0), & L_p = 1 \end{cases} \tag{16}$$

V takovém případě by c_p měla být zvolena $c_p \in \langle -1, 1 \rangle$ pro nejistou klasifikaci (možné pozadí/popředí), $c_p = K$ pro jisté popředí a $c_p = -K$ pro jisté pozadí, kde $K \gg 1$.

K může být zvoleno jako pevná, dostatečně velká konstanta, nebo lze použít definici:

$$K = 1 + \max_{p \in P_I} \sum_{q:(p,q) \in N_I} B_{(p,q)}(L_p, L_q) \tag{17}$$

Pro konkrétní klasifikaci L se A_p definuje pomocí distribuce hodnot pixelů v L:

$$A_p(L_p) = -lnP'(I_p|\theta(I, L, L_p)) \tag{18}$$

kde $\theta(I, L, L_p)$ je distribuce hodnot těch pixelů ve snímku I, které jsou v segmentaci L označeny jako L_p. Zápis $\theta(I, L, L_p)$ zjednodušíme na θ_{L_p}, když jsou I a L zřejmé z kontextu. Pravděpodobnostní funkce pro jednu hodnotu spojité veličiny nemá smysl, protože je vždy rovna nule. Původní definice $E(L)$ [14, 15] vychází z diskrétního ohodnocení pixelů obrazu, proto je potřeba P' v (18) definovat nad množinou stejně velkých disjunktních intervalů $H_1, ..., H_n$ ($\forall i, j \in \{1, ..., n\} : i \neq j \Rightarrow H_i \cap H_j = \emptyset \wedge |H_i| = |H_j|$, $\bigcup_{i \in \{1,...,n\}} H_i = H$), kde n je nejčastěji 256. Potom $P'(I_p|\theta_{L_p}) = P(H_p|\theta_{L_p})$, kde $H_p = H_i$ takové, že $I_p \in H_i$.

Výběr definice A_p závisí na charakteristice hodnot C, což závisí na jejich zdroji. Vhodný je kombinovaný přístup, kdy se definice A_p volí pro každý pixel p dle c_p:

[6] Přísně vzato, $c_p = 0$ značí neznámou klasifikaci a tedy není ani popředí, ani pozadí. Zavedením třetího typu klasifikace pixelů by se ale do systému zavedla zbytečná složitost, proto definujeme $c_p = 0$ jako pozadí.

- Pomocí c_p (16) v případě jistých hodnot.
- Pomocí L_p (18) v případě nejistého odhadu, nebo jeho absence.

Chyba $B_{p,q}$ vyjadřuje míru toho, zda pixely p a q přísluší stejnému označení a lze ji chápat jako vyjádření soudržnosti těchto pixelů. $B_{p,q}$ může být definována jako podobnost hodnot I_p a I_q, kde I_p je hodnota snímku I v p. Často se používá Gaussova funkce:

$$B_{p,q}(L_p, L_q) = ne(L_p, L_q)\frac{1}{dist(p,q)}e^{-\frac{(I_p - I_q)^2}{2\sigma_I^2}} \tag{19}$$

kde $ne(L_p, L_q) = 1$ jestliže $L_p \neq L_q$, jinak 0, $dist(p,q)$ je euklidovská vzdálenost mezi pixely a σ_I^2 je rozptyl Gaussovy funkce. Ten může být definován jako rozptyl hodnot pixelů v obraze:

$$\sigma_I^2 = \frac{1}{|N_I|}\sum_{(p,q)\in N_I}(I_p - I_q)^2 \tag{20}$$

Pro ladění segmentace se může použít váha $w_B \in \langle 0.0, 1.0\rangle$ zmírňující vliv právě definované chyby:

$$B'_{p,q}(L_p, L_q) = w_B B_{p,q}(L_p, L_q) + (1 - w_B) \tag{21}$$

kde $B'_{p,q}$ pak nahrazuje původní $B_{p,q}$ v energetické funkci E. Základní $B_{p,q}$ tedy odpovídá výchozí hodnotě váhy $w_B = 1$.

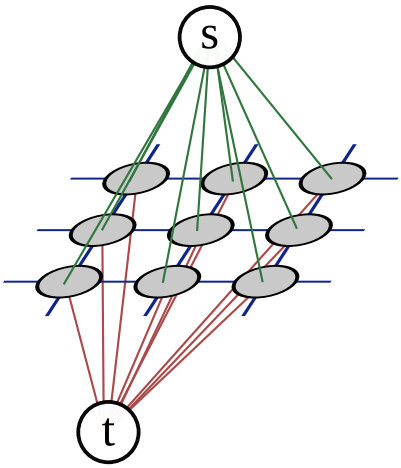
Sousednost N_I bývá nejčastěji definována čtyř-směrná, hrana existuje jen mezi vrcholy těch pixelů, které horizontálně a vertikálně bezprostředně sousedí. Rozšířená osmi-směrná sousednost bere v potaz diagonálu. Méně se používají i varianty, které spojují vrcholy pixelů ve větším okolí.

2.4.1.2 Definice grafu Problém segmentace lze převést do neorientovaného grafu[7] ze snímku a výše popsaných chyb. Graf $G = (V, E)$ je definován následovně:

- V - množina vrcholů:
  - v_p - pro každý pixel snímku $p \in P_I$ existuje jeden vrchol, který jej reprezentuje.
  - Terminály
    - s - reprezentuje popředí - segmentovaný objekt (source, zdroj).
    - t - reprezentuje pozadí (sink, stok).
- E - množina nezáporně ohodnocených hran:
  - $e_{p,q}$, $(p,q) \in N_I$ - hrany mezi všemi sousedícími vrcholy pixelů.
  - $e_{p,s}$, $e_{p,t}$ - hrany mezi všemi vrcholy pixelů a terminály.

Vizualizace grafu se čtyř-směrnou sousedností je na obrázku 14.

[7] Obecný popis této teorie používá orientovaný graf, který umožňuje přiřadit hranám různé ohodnocení (viz níže) pro různý směr. Pro většinu úloh ale dostačuje neorientovaná verze, které se dále budeme držet.



Obrázek 14: Reprezentace obrazu grafem

Ohodnocení hran vrcholů pixelů s terminály, $e_{p,s}$ a $e_{p,t}$, $p \in P_I$, jsou definovány pomocí funkce A_p:

$$e_{p,s} = A_p(0) \tag{22}$$

$$e_{p,t} = A_p(1) \tag{23}$$

Všimněme si, že hrana mezi vrcholem pixelu a terminálem popředí je ohodnocena chybou přiřazení označení pozadí danému pixelu a naopak.

Hrany mezi vrcholy pixelů jsou ohodnoceny pomocí funkce $B_{p,q}$:

$$e_{p,q} = B_{p,q}(0,1) = B_{p,q}(1,0) \tag{24}$$

Poznamenejme, že kdyby chyba $B_{p,q}$ byla definována tak, že $B_{p,q}(L_p, L_q) \neq B_{q,p}(L_q, L_p)$, bylo by nutno převést graf na orientovaný a bylo by nutno upravit definici řezu.

2.4.1.3 Grafový řez Grafový řez je rozdělení množiny vrcholů do dvou disjunktních podmnožin S a T tak, aby každá obsahovala právě jeden terminál; $s \in S$, $t \in T$. Ekvivalentně jej lze definovat jako množinu všech hran mezi vrcholy S a T, jejímž odstraněním vzniknou disjunktní podgrafy s vrcholy S a T. Minimální řez je takový, aby součet vah odstraněných hran byl minimální.

Řezem dojde k segmentaci obrazu a nalezeným objektem je množina všech pixelů snímku, jejichž vrcholy se nachází v S, $O = \{p : v_p \in S\}$. Toto odpovídá nové L takové, že

$$L_p = \begin{cases} 0, & p \in \mathrm{T} \\ 1, & p \in \mathrm{S} \end{cases} \tag{25}$$

Pro definici grafu k řezu je potřeba nějaké vstupní L; závislost $B_{p,q}$ na L je sice v grafu eliminována (24), avšak definice A_p (18) potřebuje distribuci hodnot I při segmentaci L. Toto se řeší použitím vstupu C a to buď definicí A_p přímo hodnotami C (16) nebo definicí vstupní L pomocí C:

$$L_p = \begin{cases} 0, & c_p \leq t_c \\ 1, & c_p > t_c \end{cases} \tag{26}$$

kde t_c je prahová hodnota závisející na charakteristice c_p, často $t_c = 0$. První varianta umožňuje jemněji klasifikovat pixely a vložit do systému míru jistoty, že se jedná o popředí/pozadí. Definice L pomocí C zase zjednodušuje implementaci a je vhodná když C je velmi hrubý odhad, např. když $\forall p \in P_I : c_p \approx 0 \vee c_p \approx 1$ (potom je vhodné volit $t_c = 0.5$).

2.4.1.4 Segmentace pomocí grafového řezu Vstupními parametry grafového řezu tedy jsou I - vstupní snímek a C - vstupní klasifikace pixelů obrazu I. Výstupem je O, množina pixelů definující nalezený objekt a mělo by být zjevné, že O nijak nevypovídá o typu segmentovaného objektu, jak bylo zmíněno na začátku této kapitoly.

Postup segmentace s použitím grafového řezu je tedy následující:

1. Definice L pomocí vstupního C (26).
2. Definice A_p pomocí C (16), pomocí L (18) nebo kombinovaně (viz výše).
3. Definice grafu G.
4. Řez grafu $G : S, T \rightarrow O, L'$, kde L' je segmentace dle (25).
5. Výstupem je O, segmentovaný objekt z L'

2.4.2 Varianty grafových řezů

V předchozí části byl popsán základní princip grafových řezů. Vznikly další varianty, které tuto metodu upravují ve snaze nalézt optimálnější řešení.

2.4.2.1 Grab Cut Grab Cut [16] upravuje algoritmus segmentace tak, aby byl robustnější a nalezl objekt s vyšší úspěšností při méně přesném uživatelském vstupu. V základní variantě je vstupem pouze obdélník definující oblast, kde se hledaný objekt nachází a volitelně je možno vstup rozšířit o oblasti, u kterých je jisté, že jsou součástí onoho objektu. Grab Cut je definován pro barevné (RGB) snímky, pro účely této sekce je tedy hodnota pixelu I_p definována jako třísložkový vektor.

Vstupem je množina $C' = \{c'_p : p \in P_I\}$, kde c'_p klasifikuje daný pixel třemi možnými hodnotami:

- b - jisté pozadí
- f - jisté popředí
- u - neznámá klasifikace

Aby Grab Cut dával použitelné výsledky, je potřeba, aby C' obsahovala alespoň hodnoty b a u a co nejvíce pixelů klasifikovaných jako u bylo hledaným objektem. Naopak f je volitelné.

Základní algoritmus řezu používá k výpočtu chyby A_p přímo distribuci hodnot pixelů v L (18). Grab Cut tuto distribuci nahrazuje modelem směsi gaussovských funkcí (Gaussian Mixture Model, GMM). Normální rozložení modeluje hodnoty populace pomocí jedné gaussovské funkce, která je definována střední hodnotou a rozptylem. GMM oproti tomu modeluje tyto hodnoty jako vážený součet několika gaussovských funkcí nazývané komponenty. Grab Cut používá dva modely, jeden pro popředí a jeden pro pozadí, s pevným počtem komponent K_{gmm}, typicky $K_{gmm} = 5$. Parametry GMM v tomto algoritmu lze tedy definovat jako

$$\theta_g = \{\pi(l,k), \mu(l,k), cov(l,k) : l \in \{0,1\}, k \in \{1,2,...,K_{gmm}\}\} \tag{27}$$

kde komponenta k modelu popředí ($l = 1$) a pozadí ($l = 0$) je definována parametry $\pi(l,k)$, nezáporná váha ($\forall l : \sum_k \pi(l,k) = 1$), střední hodnota $\mu(l,k)$ a kovarianční matice $cov(l,k)$ (rozptyl, pokud by hodnota pixelu byl skalár).

Dále je zavedena množina $K_g = \{k_p : p \in P_I, k_p \in \{1,2,...,K_{gmm}\}\}$, která pixelu p přiřazuje komponentu k_p z modelu popředí, pokud $L_p = 1$, nebo pozadí, když $L_p = 0$.

Grab Cut definuje funkci $D_p(L_p, k_p, \theta_g)$, která počítá chybu při přiřazení klasifikace L_p a komponenty k_p pixelu p při konkrétním modelu θ_g:

$$\begin{aligned} D_p(L_p,k_p,\theta_g) = &- \ln p(I_p|L_p,k_p) - \ln \pi(L_p,k_p) = \\ &- \ln \pi(L_p,k_p) + \frac{1}{2}\ln(C_g\, det(cov(L_p,k_p))) + \\ &\frac{1}{2}[I_p - \mu(L_p,k_p)]^T cov(L_p,k_p)^{-1}[I_p - \mu(L_p,k_p)] \end{aligned} \tag{28}$$

kde $C_g = (2\pi)^{K_{gmm}}$ je konstanta, kterou [16] vynechává.

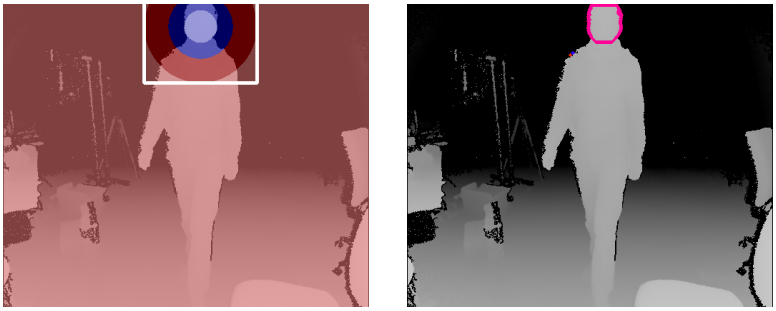
Energetická funkce je upravena na

$$E(L,K_g,\theta_g) = A(L,K_g,\theta_g) + B(L) = \sum_{p \in P_I} A_p(L_p,k_p,\theta_g) + \sum_{(p,q)\in N_I} B_{p,q}(L_p,L_q) \tag{29}$$

kde chyba A_p je definována pomocí D_p:

$$A_p(L_p,k_p,\theta_g) = D_p(L_p,k_p,\theta_g) \tag{30}$$

Postup segmentace je následující:



(a) Vstupní maska (b) Výstupní segmentace

Obrázek 15: Příklad výstupu Grab Cut; použitá implementace používá u_f (nejisté popředí) a u_b (nejisté pozadí) místo u. Červená barva znázorňuje pozadí, modrá popředí. Vnitřní kruh je jistá klasifikace, mezikruží nejistá klasifikace a prostor vně kružnic je jisté pozadí.

1. Definice množiny L ze vstupu C': $L_p = 1$ když $c'_p = u \vee c'_p = f$, jinak $(c'_p = b)$ $L_p = 0$.
2. Výpočet parametrů GMM θ_g pro popředí z $\{I_p : p \in P_I \wedge L_p = 1\}$ a pozadí z $\{I_p : p \in P_I \wedge L_p = 0\}$.
3. Definice K_g pro $\{p : c'_p = u\}$:

$$k_p = \min_{k_p} D_p(L_p, k_p, \theta_g) \tag{31}$$

   k_p lze nalézt pomocí prostého výpočtu D_p pro všechny k.

4. Výpočet nových parametrů θ_g na základě K_g. Definujme $F(l,k) = \{I_p : p \in P_I, L_p = l, k_p = k\}$, množina všech pixelů, které jsou klasifikovány jako $l \in L$ a K_g jim přiřazuje komponentu $k \in \{1, 2, ..., K_{gmm}\}$.

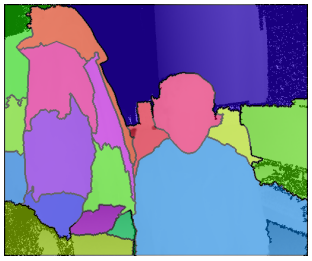
   Potom $\mu(l,k)$ je střední hodnota $F(l,k)$, $cov(l,k)$ je kovariance hodnot $F(l,k)$ a $\pi(l,k)$ je váha:

$$\pi(l,k) = \frac{|F(l,k)|}{\sum_{k_i \in \{1,2,...,K_{gmm}\}} |F(l,k_i)|} \tag{32}$$

5. Řez grafu minimalizací $E(L, K_g, \theta_g)$ (29) a aktualizace L pro $p : c'_p = u$, tedy klasifikace pixelů, které jsou označeny jako jisté popředí a pozadí se nemění.
6. Opakovat od kroku 3, dokud není dosažena konvergence.

Příklad segmentace Grab Cut je na obrázku 15.

2.4.2.2 One Cut One Cut rozšiřuje graf, aby se eliminovala potřeba iterace Grab Cut a zlepšila kvalita detekce[17] přidáním pomocných vrcholů, které spojují vrcholy pixelů s podobnými hodnotami.



Obrázek 16: Příklad výstupu Watershed

Interval hodnot pixelů $H = \langle h_{min}, h_{max} \rangle$ se rovnoměrně rozdělí do k intervalů B_1, B_2,... B_k. Pro každý interval je vytvořen korespondující pomocný vrchol grafu b_i. Pro každý vrchol pixelu p je vytvořena hrana e_{p,b_i} s takovým pomocným vrcholem b_i, že $I_p \in B_i$, tj. hodnota pixelu p je definována v intervalu B_i reprezentovaný tímto pomocným vrcholem.

Váha hrany s pomocným vrcholem je v nejjednodušším případě rovna konstantě β (většinou $\beta \in \langle 0.5, 1 \rangle$), která definuje váhu vlivu pomocných vrcholů. [17] dále definuje výpočet β na základě distribuce hodnot v oblasti vstupní klasifikace pixelů a váhy β' v případě komplexnějších typů snímků.

2.5 Segmentace obrazu metodou Watershed

Watershed [18, 19] přistupuje k obrazu, jako by se jednalo o reliéf, kdy světlé pixely reprezentují vrcholy a tmavé pixely jsou prohlubně. Algoritmus tedy nepracuje přímo s barevným snímkem, ale nejprve spočítá gradient (první derivaci). Systém začíná vyplňováním prohlubní (lokální minima) a rozšiřuje je, dokud nedosáhnou hranic s jinou prohlubní. Hranice mezi jednotlivými minimy pak definují segmentaci.

Výsledky takového postupu ovšem obvykle naleznou velké množství malých oblastí - každá pro jedno lokální minimum - a taková segmentace je prakticky nepoužitelná. Proto Watershed nezačíná ve všech lokálních minimech, ale vyžaduje jako vstup markery definující počet a odhadnutou pozici segmentovaných objektů.

Tato metoda tedy zjevně není založena na grafových řezech a v této práci je zahrnuta pro srovnání s alternativním způsobem segmentace. Příklad výstupu je na obrázku 16.

2.6 Klasifikace objektů

Segmentovaný objekt O je jen množina bodů, která nenese žádnou přímou informaci o typu objektu. Proto je potřeba určit míru toho, zda se jedná o lidskou hlavu.

Klasifikace je provedena pomocí kombinace výsledků dílčích chyb $e'_i(I, O) \geq 0$. Každá chyba $e'_i(O, I)$ měří rozdíl nějakého aspektu objektu O ve snímku I od očekávaného modelu, např. tvar hrany O. Jednotlivé dílčí chyby jsou popsány v následujících sekcích.

Celková chyba je definována jako

$$e(I,O) = \sum_i w_{e'_i}\, e'_i(I,O) \tag{33}$$

kde $w_{e'_i} \in \langle 0,1 \rangle$ je váha dílčí chyby e'_i. Objekt je klasifikován jako lidská hlava, jestliže platí:

$$(\forall i : e'_i(I,O) \leq t_{e'_i}) \wedge (e(I,O) \leq t_e) \tag{34}$$

kde $t_{e'_i}$ je prahová hodnota chyby e'_i a t_e je prahová hodnota výsledné chyby e.

Parametry klasifikace jsou tedy prahy $T_e = \{t_e, t_{e'_1}, t_{e'_2}, ...\}$ a váhy $W_e = \{w_{e'_1}, w_{e'_2}, ...\}$. Vstupem je objekt O a obraz I, výstupem je dvojice (e,r), kde e je celková chyba $e(I,O)$ a r je 1 jestliže platí (34), jinak O.

2.6.1 Aproximace tvaru objektu elipsou

Obrys lidské hlavy lze aproximovaně modelovat elipsou, čehož využívají některé dílčí algoritmy. Definujme tedy elipsu $El(O) = (s,a,b,\alpha)$, kde $s = (x,y)$ je střed, a, b jsou hlavní a vedlejší osy a α je úhel hlavní osy. Parametry jsou zvoleny tak, aby chyba E_{El} byla minimální:

$$E_{El}(O) = \sum_{p \in Co(O)} \delta(El(O), p) \tag{35}$$

kde $\delta(El(O),p)$ je vzdálenost bodu p od elipsy $El(O)$ a $Co(O)$ je kontura O:

$$Co(O) = \{p : (p \in O) \wedge [\exists q \in (P_I \setminus O) : (p,q) \in N_I]\} \tag{36}$$

Dále definujme $El'(O)$ jako množinu bodů $p = (x_p, y_p)$ v souřadném systému snímku I, kde $x_p, y_p \in Z$ a p leží v elipse $El(O)$. Body elipsy mohou mít souřadnice mimo obraz I, minimalizace E_{El} umožňuje nalézt takovou elipsu.

Definice $\delta(El(O),p)$ a problém minimalizace E_{El} jsou řešené v [20], kde je navrženo několik algoritmů. Většina z nich vychází z funkce $F(k,p)$:

$$F(k,p) = K_{xx}x^2 + K_{xy}xy + K_{yy}y^2 + K_x x + K_y y + K_0 \tag{37}$$

kde $k = (K_{xx}, K_{xy}, K_{yy}, K_x, K_y, K_0)$ jsou parametry definující kuželosečku - elipsa je jeden z typů kuželoseček - a $p = (x,y)$ je bod ležící křivce, kterou parametry K_i definují. $F(k,p)$ lze vyjádřit pomocí skalárního součinu:

$$F(k,p) = (x^2, xy, y^2, x, y, 1) \cdot (K_{xx}, K_{xy}, K_{yy}, K_x, K_y, K_0) \tag{38}$$

Chyba je pak definována jako

$$E_{El}(O) = \sum_{p \in Co(O)} F(k,p)^2 = |Dk|^2 \tag{39}$$

kde D je matice složená z hodnot $(x^2, xy, y^2, x, y, 1)$ ze všech $p \in Co(O)$. Problém minimalizace $E_{El}(O)$ dle (39) je pak definován jako

$$D^T Dk = \lambda k \Leftrightarrow D^T Dk - \lambda k = 0 \tag{40}$$

kde řešením je vlastní vektor $D^T D$ pro nejmenší vlastní číslo. Protože pro konkrétní vlastní číslo existuje nekonečně mnoho vlastních vektorů, je dále zavedena podmínka $|k| = 1$.

Tato metoda je velmi citlivá na šum a odlehlá pozorování a tedy v případě zašumělého vstupu dává horší výsledky. Proto vnikly rozšiřující metody, které se tento problém snaží zmírnit [20].

Parametry k popisují obecnou kuželosečku, výsledkem tedy nemusí být elipsa. Např. implementace v OpenCV [21] toto řeší následujícím postupem:

- Odhad středu elipsy s' průměrem všech vstupních bodů.
- Výpočet k z bodů posunutých o $-s'$.
- Výpočet středu kuželosečky s z parametrů k.
- Výpočet $k' = (K'_{xx}, K'_{xy}, K'_{yy})$ z bodů posunutých o $-s$.
- Výpočet ostatních parametrů elipsy z k'.

2.6.2 Porovnání s plochou elipsy

Jak již bylo zmíněno, obrys lidské hlavy lze aproximovaně modelovat elipsou. Je tedy záhodno určit míru podobnosti nalezené elipsy a segmentovaného objektu.

$$e'_{ellipseArea}(I,O) = \frac{|O \cup El'(O)| - |O \cap El'(O)|}{|El'(O)|} \tag{41}$$

definuje chybu plochy elipsy a segmentovaného objektu normalizováno plochou elipsy.

2.6.3 Poměr poloměrů elipsy

Nalezená elipsa může mít různé tvary, ale rozměry lidské hlavy se pohybují ve velmi omezených hodnotách[22]. Poměr $r = \frac{b}{a}$ lze tedy využít k velmi jednoduchému odmítnutí objektů, jejichž tvar je příliš protáhlý.

$$e'_{ellipseRadiusRatio}(I,O) = \frac{\max(r_{min} - r, 0) + \max(r - r_{max}, 0)}{r_{max} - r_{min}} \tag{42}$$

definuje chybu tohoto poměru, kde $r = \frac{b}{a}$ $(El(O) = (s, a, b, \alpha), a \geq b \Rightarrow r \geq 1)$ a r_{min} a r_{max}, $r_{min} < r_{max}$ jsou konstanty reprezentující očekávaný rozsah poměru hlavního a vedlejšího poloměru elipsy.

2.6.4 Hloubkový tvar

Doposud představené dílčí chyby berou v potaz pouze O, body segmentovaného objektu. Velmi důležitou informaci ale nese samotný snímek I, tedy hodnoty pixelů v objektu $\{I_p : p \in O\}$. Jako zjednodušený model byl zvolen elipsoid.

Výpočet dílčí chyby z hloubkového tvaru se skládá ze dvou kroků; nejprve je potřeba nalézt vhodné parametry elipsoidu, po té jej porovnat se skutečným obrazem.

Elipsoid definujeme jako $T(O) = (s, a, b, c, m, \alpha)$, kde $s = (x_s, y_s)$ je střed v souřadném systému snímku, a, b jsou poloměry v rovině obrazu, c je třetí poloměr, kolmý na tuto rovinu a tedy odpovídá hloubce, α je úhel osy a a m je hloubka středu s. Parametry s, a, b, α tedy odpovídají parametrům dříve nalezené elipsy $El(O)$ a zůstávají neznámé m, c. Ty nalezneme minimalizací chyby

$$E_T(O) = \sum_{p \in O} (T_p(O) - I_p)^2 \tag{43}$$

kde $T_p(O)$ je hodnota hloubky elipsoidu $T(O)$ v bodě p (v souřadném systému snímku):

$$T_p(O) = m + c\, T'_p(O) \tag{44}$$

$$T'_p(O) = \sqrt{1 - Clamp_{01}(d_p)^2} \tag{45}$$

$$d_p = \frac{\sqrt{(a\, y'_p)^2 + (b\, x'_p)^2}}{a\, b} \tag{46}$$

$$(x'_p, y'_p, 1) = (x_p, y_p, 1) M_T \tag{47}$$

$$M_T = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_s & -y_s & 1 \end{pmatrix} \begin{pmatrix} cos(-\alpha) & sin(-\alpha) & 0 \\ -sin(-\alpha) & cos(-\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{48}$$

Matice M_T (48) definuje transformaci bodu snímku $p = (x_p, y_p)$ do bodu v souřadné soustavě elipsy $p' = (x'_p, y'_p)$ prostřednictvím rovnice (47). První část definice matice je inverzní translací středu elipsy $s = (x_s, y_s)$ - posune počátek do systému elipsy - druhá část definice matice invertuje rotaci elipsy α. d_p je relativní vzdálenost bodu p od středu elipsy vzhledem k bodu na elipse ve směru ps; výpočetní vztah (46) vychází z matematické definice elipsy $\frac{x^2}{a^2} + \frac{y^2}{b^2} = d^2 = 1$.

$T'_p(O)$ definuje relativní hodnotu hloubky elipsoidu v bodě p; vztah (45) počítá bod na jednotkové kružnici ($x^2 + y^2 = 1 \Rightarrow y = \sqrt{1 - x^2}$), který je výpočtem (44) převeden na hloubku elipsoidu v I. $Clamp_{01}(d_p)$ je definován jako $max(0, min(1, d_p))$.

Minimalizaci (43) lze provést metodou nejmenších čtverců, tedy vyřešením soustavy rovnic

$$\frac{\partial E_T(O)}{\partial m} = \sum_{p \in O} 2(m + c\,T'_p(O) - I_p) = 0 \tag{49}$$

$$\frac{\partial E_T(O)}{\partial c} = \sum_{p \in O} 2\,T'_p(O)(m + c\,T'_p(O) - I_p) = 0 \tag{50}$$

Po vyřešení jsou k dispozici výpočetní vztahy pro m, c:

$$c = \frac{sum_{I_p T'_p}|O| - sum_{I_p} sum_{T'_p}}{sum_{T'^2_p}|O| - sum^2_{T'_p}} \tag{51}$$

$$m = \frac{sum_{I_p} - c\,sum_{T'_p}}{|O|} \tag{52}$$

kde

$$sum_{I_p} = \sum_{p \in O} I_p \tag{53}$$

$$sum_{T'_p} = \sum_{p \in O} T'_p(O) \tag{54}$$

$$sum_{T'^2_p} = \sum_{p \in O} T'_p(O)^2 \tag{55}$$

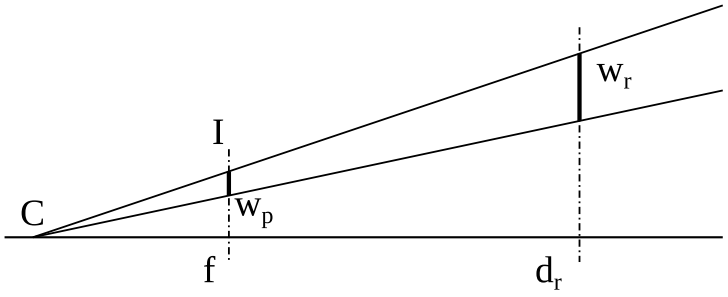
$$sum_{I_p T'_p} = \sum_{p \in O} I_p T'_p(O) \tag{56}$$

Dílčí chybu hloubkového obrazu lze definovat jako

$$e'_{depthShape}(I, O) = \frac{E_T(O)}{c^2|O|} \tag{57}$$

Nalezené parametry dále vypovídají o dalších vlastnostech O v I; m definuje hloubku, v níž se nalezený objekt nachází a c popisuje tvar nalezeného objektu. Pokud c vyjde záporné, což dle (51) je možné, vypovídá o vydutém (konkávním) tvaru segmentovaného objektu. Takový zcela jistě nemůže být lidskou hlavou a tedy podmínku (34) lze rozšířit o $(c > 0)$[8].

[8] Pokud by interpretace hodnot obrazu I byla inverzní, tj. h_{min} reprezentovalo blízké body a h_{max} vzdálené, pak by celý tento vztah byl taktéž invertovaný a podmínka konvexního tvaru by byla $c < 0$.



Obrázek 17: Model hloubkové kamery a promítání reálného objektu do snímku

2.6.5 Velikost objektu

Další informací, kterou lze z hloubkového obrazu získat, je velikost samotného objektu. To je ale možné jen v případě, že jsou známé vnitřní parametry kamery, které umožňují přepočítat hodnotu hloubky do fyzických rozměrů. Tyto parametry by měly být k dispozici v dokumentaci použitého zařízení a protože se vztahují k původním hodnotám pixelů snímku, v této části jsou hodnoty $I_p \in I$ uvažovány v originální, nenormalizované podobě.

Pro výpočet velikosti segmentovaného objektu lze použít model elipsoidu $T(O)$ popsaném výše; a (hlavní osa elipsy - řez elipsoidu v rovině rovnoběžné s rovinou obrazu) definuje výšku (v pixelech) a m, hloubka středu elipsoidu, definuje vzdálenost objektu od kamery. Podobně jako I_p, i m je v následující části uvažován v hodnotě odpovídající původní interpretaci pixelů závisející na zdroji hloubkových obrazů. Lze získat z vypočtené hodnoty jednoduše inverzí výpočtu normalizace snímku, která byla popsána dříve.

Pro model hloubkové kamery je použita dírková komora. Model systému kamery, nalezeného objektu a jeho reálné podobny je znázorněn na obrázku 17.

C reprezentuje střed promítání hloubkové kamery, I je hloubkový snímek, který zároveň definuje rovinu promítání, w_r $[m]$ je výška reálného objektu, který se nachází ve vzdálenosti d_r $[m]$ od kamery, w_p $[px]$ je výška tohoto objektu promítnutá v obraze I a f $[m]$ je ohnisková vzdálenost kamery.

Nechť p $[\frac{px}{m}]$ je vnitřním parametrem hloubkové kamery, který vyjadřuje rozměr pixelu v rovině promítání. Potom platí

$$w'_p = \frac{w_p}{p} \tag{58}$$

kde w'_p $[m]$ je fyzický rozměr objektu ve snímku I v rovině promítání. Dále definujme přepočet hloubky snímku na hloubku v prostoru jako

$$d_r = I_p\, z + c \tag{59}$$

kde z $[\frac{m}{d}]$ a c $[m]$ jsou další vnitřní parametry hloubkové kamery a $[d]$ je jednotka hloubky v obraze, ve které jsou vyjádřeny hodnoty pixelů I_p.

Z podobnosti trojúhelníků a modelu na obrázku 17 vyplývá vztah

$$\frac{w_r}{w'_p} = \frac{d_r}{f} \tag{60}$$

který lze upravit

$$w_r = w'_p \frac{d_r}{f} = \frac{w_p}{p} \frac{I_p\, z + c}{f} \tag{61}$$

Tato rovnice tedy definuje, jak lze z hodnoty hloubky v obraze I_p a rozměru objektu ve snímku w_p v rovině rovnoběžné s rovinou promítání získat skutečný rozměr w_r za pomoci parametrů (z, c, f, p). Parametry f a p lze nahradit

$$f' = p\, f \tag{62}$$

a tedy konečnými potřebnými parametry jsou (z, c, f') a výpočetní vztah pro rozměr objektu je

$$w_r = \frac{1}{f'} w_p (I_p\, z + c) \tag{63}$$

Chybu lze pak definovat jako

$$e'_{realSize}(I, O) = \frac{\max(w_{min} - w_r, 0) + \max(w_r - w_{max}, 0)}{w_{max} - w_{min}} \tag{64}$$

kde w_{min} a w_{max} $[m]$ definují očekávaný interval hodnot rozměru w_r.

Podle [22] výška hlavy u 98% lidí je v rozsahu 19,8 cm - 25,5 cm. Po zvážení odchylek a pokrývky hlavy byl pro měření chyby zvolen rozsah 18 cm - 35 cm. Výrazně vyšší horní hranice byla zvolena i proto, že hodnoty v [22] jsou měřeny v profilu, avšak v hloubkových snímcích je nutno uvažovat různé směry pohledu na lidskou hlavu, kdy viditelný rozměr může být pod určitým úhlem větší, jako je např. šikmo shora.

2.7 Sledování objektu mezi snímky

Úkolem tohoto kroku je optimalizace detekce tak, že nalezenou hlavu z předchozího snímku se pokusí vyhledat v obraze dalším a díky tomu se v ideálním případě bude segmentovat jen jeden marker (výstup sledování objektu mezi snímky) místo množiny markerů z detekce potenciálně zajímavých objektů.

2.7.1 Vyhledávání pomocí šablony

Vyhledávání podobných objektů pomocí šablony je popsáno v [23]. Vstupem tohoto algoritmu je šablona T - snímek, o rozměrech w_T, h_T, které jsou menší, než w_I, h_I, rozměry obrázku I, v němž objekt vyhledáváme. Výstupem je S, matice o rozměrech $w_S = w_I - w_T + 1$ a $h_S = h_I - h_T + 1$,

která pozici $p \in P_S$ přiřazuje číslo $S_p \in R$, které vyjadřuje podobnost objektu T ve snímku I v oblasti definované levým horním rohem p a rozměry w_T, h_T. P_S je množina všech bodů S a P_T je množina všech bodů T.

S_p může být definováno různými výpočetními vztahy podobnosti ([23]), např.

$$S_p = \sum_{q \in P_T} (T_q - I_{p+q})^2 \tag{65}$$

$$S_p = \frac{\sum_{q \in P_T} (T_q - I_{p+q})^2}{\sqrt{\sum_{q \in P_T} T_q^2 \sum_{q \in P_T} I_{p+q}^2}} \tag{66}$$

kde (66) je normalizovaná varianta (65).

V případě hloubkových obrazů definujeme šablonu T jako podobraz snímku I definovaný minimálními a maximálními souřadnicemi bodů v nalezeném objektu O.

Hledaný objekt nalezneme určením minimální nebo maximální hodnoty S_p; některé vztahy podobnosti v [23] vracejí nejlepší shodu ve formě maxima, jiné jako minimum. Proto dále definujeme S', normalizovanou S, jako

$$S'_p = \begin{cases} S_p, & \text{pokud vyšší hodnota } S_p \text{ vyjadřuje větší shodu} \\ -S_p, & \text{pokud vyšší hodnota } S_p \text{ vyjadřuje větší chybu} \end{cases} \tag{67}$$

V případě, že se v S vyskytne více hodnot blízkých maximální hodnotě, je vhodné preferovat pozice bližší originálnímu objektu na základě dříve zmíněného předpokladu, že objekt svou pozici mění mezi snímky v malé míře. Tuto preferenci je možno vyjádřit úpravou hodnot S'_p:

$$S''_p = S'_p - dist(p, p_o) \frac{1}{\sqrt{(w_I^2 + h_I^2)}} w_{tm} \tag{68}$$

kde p_o je původní pozice šablony T, $dist$ je euklidovská vzdálenost a w_{tm} je váha, která určuje míru důrazu na preferenci objektů blíže p_o.

Objekt je v novém snímku nalezen, jestliže pro maximální S''_p platí $S''_p \geq t_{tm}$, kde t_{tm} je prahová hodnota, parametr vyhledávání. Výstupním markerem je pak $O' = \{q' : q' = (q + p - p_o), q \in O\}$.

2.7.2 Vyhledávání pomocí význačných bodů a párování

Tato metoda [24] se skládá z několika kroků a je velmi často používána při zpracováni světelných obrazů, k vyhledávání známých objektů ve scéně. Postup je následující:

- Nalezení význačných bodů (features) - v celém snímku se vyhledají takové pozice, které jsou velmi výrazné.
- Popis význačných bodů (description) - k nalezeným bodům jsou na základě hodnot pixelů v okolí přiřazeny vektory, které co nejlépe popisují tento bod na objektu.

- Párování (matching) - jestliže jsou k dispozici dvě množiny popsaných význačných bodů ze dvou snímků, párování je metoda, kdy se naleznou takové dvojice bodů z těchto množin, které s co nejvyšší pravděpodobností odpovídají stejné části stejného objektu.

Nalezené spárované body vypovídají o změně pozice objektu mezi snímky. Lze je využít k nalezení transformace H, se kterou lze ze vstupní kontury $Co(O)$ vypočíst výstupní, nově nalezenou konturu $Co'(O)$, již lze využít k definici markeru reprezentujícího nalezenou hlavu v novém snímku.

2.7.2.1 Nalezení význačných bodů Bylo navrženo několik algoritmů k řešení tohoto problému, jedním z nich je Harrisův detektor rohů [25], který hledá takové body, kdy se intenzita pixelů mění ve všech směrech (oproti hranám a plochým oblastem). Tato podmínka je vyjádřena funkcí:

$$E(u,v) = \sum_{(x,y)\in P_I} w(x,y)(I(x+u,y+v) - I(x,y))^2 \tag{69}$$

kde (u,v) je posun, pro který je počítána změna intenzity ve všech směrech, w definuje oblast, pro kterou je funkce počítána; je určena středem - potenciálním rohem - a jeho okolím, které může být váženo Gausiánem nebo vrací 1 pro okolí, jinak 0. Detekce rohů je pak definována jako maximalizace $E(u,v)$, čehož lze dosáhnout maximalizací druhého činitele v (69).

Aplikováním taylorova rozvoje lze $E(u,v)$ vyjádřit jako

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} \tag{70}$$

kde

$$M = \sum_{(x,y)\in P_I} w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix} \tag{71}$$

kde I_x a I_y jsou aproximací derivace obrazu ve směrech x a y. Na základě těchto úprav je definována míra toho, zda je okolí bodu rohem jako

$$R = det(M) - k(trace(M))^2 \tag{72}$$

kde $det(M) = \lambda_1\lambda_2$, $trace(M) = \lambda_1 + \lambda_2$, λ_1 a λ_2 jsou vlastními čísly M a $k \in [0.04, 0.06]$ je empiricky nalezená konstanta [26].

Oblast w je rohem, jestliže je R velké číslo. Je-li R malé, jedná se o plochou oblast a hranu indikuje $R < 0$. Detekce rohů pak spočívá ve výpočtu R pro okolí všech bodů I a nalezení hodnot, které jsou dostatečně velké.

2.7.2.2 Popis význačných bodů Taktéž k popisu význačných bodů vzniklo několik různých algoritmů. Metoda BRIEF [27] byla navržena za účelem rychlejšího vytvoření a párování vektorů popisující body ve srovnání s jinými populárními algoritmy, jako je např. SURF. Tato metoda definuje testovací funkci

$$\tau(w, p, q) = \begin{cases} 1 & \text{pokud } w(p) < w(q) \\ 0 & \text{v ostatních případech} \end{cases} \tag{73}$$

kde w je okolí popisovaného bodu v I (pod-obraz I) filtrovaného Gausiánem a p, q jsou souřadnice bodů tohoto okolí.

Okolí w je pak popsáno bitovým řetězcem

$$f_n(w) = \sum_{1 \leq i \leq n} 2^{i-1} \tau(w, p_i, q_i) \tag{74}$$

kde $(p_i, q_i) \in S$ jsou páry souřadnic bodů, které jsou porovnány. Konfiguračními parametry tohoto algoritmu jsou tedy n, velikost popisného vektoru, parametr Gausiánu σ pro rozostření w, rozměry oblasti w a páry porovnávaných souřadnic bodů S.

Podle [27] jsou vhodnými parametry

- $n = 128, 256$ a 512
- $\sigma = 2.0$
- $s = 9$, kde s je šířka a výška oblasti w
- S - náhodně vybrané dvojice bodů v oblasti w s normálním rozložením $\mu = 0$ a $\sigma^2 = \frac{1}{25} s^2$

ORB [28] je algoritmus založený na BRIEF, který je invariantní vůči rotaci a dále optimalizovaný.

2.7.2.3 Párování význačných bodů Popis každého význačného bodu je vektor f, párování je tedy postup, jehož výsledkem je nalezení dvojic ze dvou množin F_1, F_2 těchto vektorů, jejichž vzdálenost je nejmenší. Definice vzdálenosti záleží na charakteristice a zdroji hodnot f - např. v případě BRIEF a ORB popsaných výše je vhodná Hammingova vzdálenost.

Jestliže jsou množiny F_1 a F_2 relativně malé, je možno použít hrubou sílu, tedy výpočet vzdálenosti všech párů. V případě, kdy jsou množiny výrazně větší, výpočetní náročnost adekvátně stoupá, proto vznikly techniky rychlejšího vyhledávání v n-dimenzionálním prostoru, např.:

- kd-strom (k-dimenzionální strom) - binární vyhledávací strom, který je tvořen tak, že každý uzel rozděluje prostor na dvě části. Dimenze dělení se obvykle volí ta s největším rozptylem a hodnota bývá medián.

- Hierarchický k-means strom - množina vektorů je rozdělena do K shluků metodou k-means. Každý z těchto shluků je pak dále opět dělen do K shluků rekurzivně, dokud je velikost shluku větší, než K.

- FLANN [29] - metoda, která využívá několika různých algoritmů a vybere jeden z nich a jeho parametry na základě reprezentativní množiny vektorů a požadovaných vlastností (rychlost konstrukce vyhledávací struktury, paměťová náročnost, a další). Tato metoda minimalizuje výpočetní náročnost za cenu určité nepřesnosti - nalezený vektor nemusí být nejbližší k hledanému, většinou ale bude blízko.

2.7.2.4 Transformace mezi snímky Po provedení párování jsou k dispozici nejen dvojice popisných vektorů (f_1, f_2), ale také jejich původní souřadnice v I, (p_1, p_2).

Vzhledem k šumu a změnám mezi snímky nelze očekávat, že bude nalezeno dost bodů v novém snímku, aby dostatečně reprezentovaly oblast, která je vyhledávána na základě dříve detekované hlavy. Páry bodů (p_1, p_2) lze ale použít k výpočtu transformace bodů z původního snímku do nového snímku podle změny pozice spárovaných bodů:

$$\begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} = H \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{75}$$

kde (x_i, y_i) je bod v původním snímku, (x'_i, y'_i) je bod ve snímku novém a H je transformační matice. H lze získat vyřešením soustavy rovnic (75) pro spárované body; tato soustava je naddimenzovaná, tedy je potřeba nalézt aproximované řešení minimalizující chybu

$$E(H) = \sum_i ((x'_i - x''_i)^2 + (y'_i - y''_i)^2) \tag{76}$$

kde (x''_i, y''_i) je transformovaný $p_{1i} = (x_i, y_i)$ podle (75), $(x'_i, y'_i) = p_{2i}$ a (p_{1i}, p_{2i}) je pár bodů z podobnými popisnými vektory.

Prakticky se dá očekávat, že mezi páry existují odlehlá pozorování. K jejich nalezení a vyloučení lze použít nedeterministickou metodu RANSAC [30], jejíž postup při aplikování na problém hledání H je následující:

- Náhodně se vybere podmnožina dvojic bodů a z nich se vypočte H.

- Pro každou dvojici bodů (p_1, p_2) je spočten bod p'_2 pomocí H a je-li rozdíl mezi p_2 a p'_2 větší, než určitá prahová hodnota, je tato dvojice považována za odlehlé pozorování.

- Tento postup je zopakován k-krát. Z výsledků, jejichž počet dvojic zhodnocených jako odlehlé pozorování je menší než limitní hodnota, je vybrána H s nejmenší chybou (76).

S nalezenou H lze pak pomocí (75) původní konturu $Co(O)$ reprezentující dříve detekovanou hlavu transformovat do nové kontury $Co'(O)$ a tu použít jako marker pro segmentaci.

3 Implementace

Implementace byla realizována v programovacím jazyce C++ s využitím standardu C++11, kompilátoru gcc 5.4.0 v OS Ubuntu 16.04 64 bit. Použité knihovny třetích stran jsou:

- OpenCV 2.4.9 - pro většinu operací zpracování obrazu, načítání a ukládání snímků.
- Maxflow [31] - implementace graph cut a one cut.
- Inih [32] - načítání konfiguračních ini souborů.
- YAML [33] - načítání a ukládání souborů v textovém formátu yaml.
- Pthreads [34] - podpora vícevláknového programování.

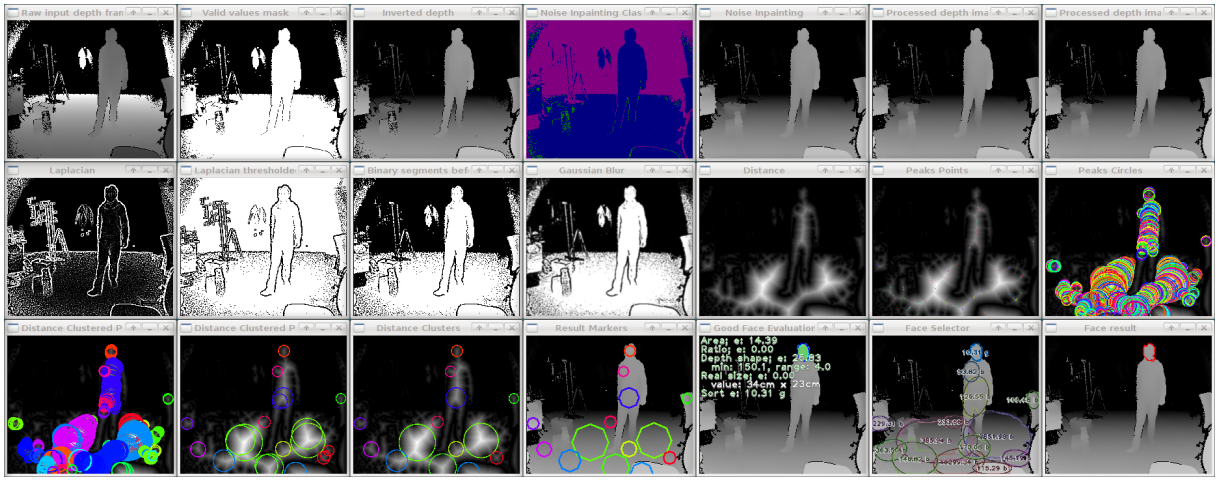
3.1 Funkcionalita aplikace

Aplikace umožňuje spustit detekci ve třech různých módech pro odlišné účely. Parametry jednotlivých kroků detekce a segmentace je možno nastavit v konfiguračním souboru ve formátu ini. Tuto konfiguraci je možno upravit ve formě argumentů předaných při spuštění přes příkazovou řádku. Implementované úlohy jsou:

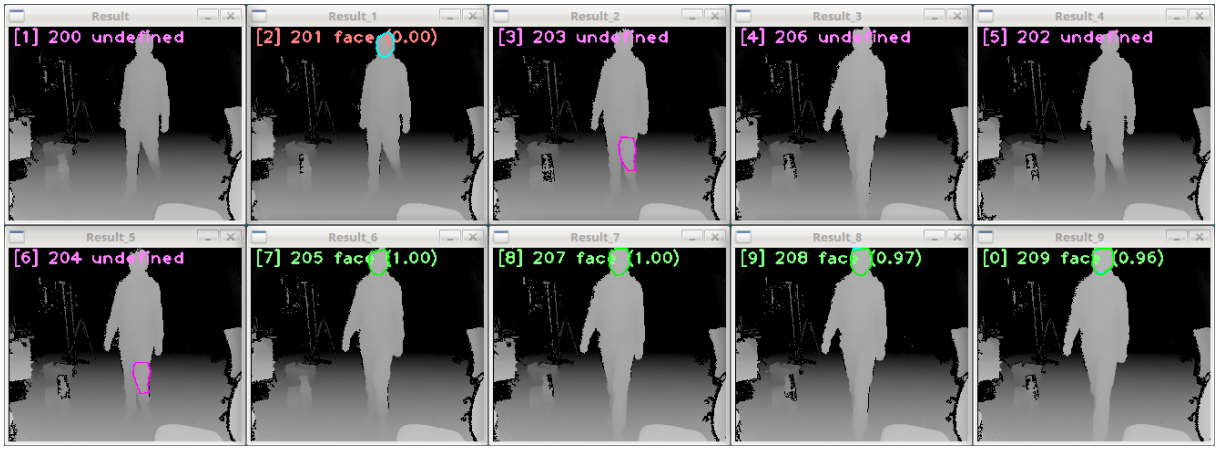
- Detekce hlavy - postupně zpracovává snímky vstupního videa a uživateli zobrazuje jednotlivé kroky detekce při dané konfiguraci. V průběhu této úlohy je možno:
  - Přehrávat jednotlivé snímky nebo krokovat dle klávesového vstupu.
  - Uložit zobrazené kroky detekce do souborů.

  Příklad grafického výstupu této úlohy je na obrázku 18.
- Výběr správných výsledků - postupně zobrazuje zpracované snímky po deseti a uživatel pomocí klávesového vstupu klasifikuje jednotlivé výsledky; zda je aktuálně nalezená hlava správná nebo zda snímek žádnou neobsahuje. Tyto informace jsou poté uloženy na disk a je možno je využít v jiných úlohách, především v měření kvality detekce - uložená data jsou použita k porovnání detekce s jinými parametry a určení správnosti výsledku.

  Příklad grafického výstupu je na obrázku 19. Při této úloze je vhodné použít kaskádový klasifikátor Haar pro zvýšení přesnosti detekce.
- Měření kvality detekce - aplikace nemá žádný grafický výstup a automaticky postupně zpracuje všechny snímky. Do souboru uloží výsledky zpracování; jak dlouho trvalo zpracování jednotlivých obrázků a kvalita detekce, k čemuž je použita klasifikace z předchozího typu úlohy.



Obrázek 18: Příklad výstupu úlohy detekce hlavy



Obrázek 19: Příklad výstupu úlohy výběr správných výsledků

3.2 Architektura implementace

Kód je rozdělený do několika modulů:

- Core - pomocné obecné funkce a třídy, které se netýkají detekce hlavy a zpracování obrazu obecně.
- INI - rozhraní nad knihovnou Inih. Slouží k načítání konfiguračního ini souboru.
- YAML - rozhraní nad knihovnou YAML. Slouží k načítání a ukládání souborů tohoto formátu.
- OpenCVUtility - pomocné funkce rozšiřující knihovnu OpenCV.
- Profiler - profilování aplikace; měření časové a paměťové náročnosti algoritmů.
- DepthFaceDetectorLib - implementace detekce hlavy v hloubkových obrazech, jak je popisováno v tomto dokumentu. Jedná se o množinu tříd a funkcí implementující tyto algoritmy.
- DepthFaceDetector - aplikace poskytující výše popsané úkony.

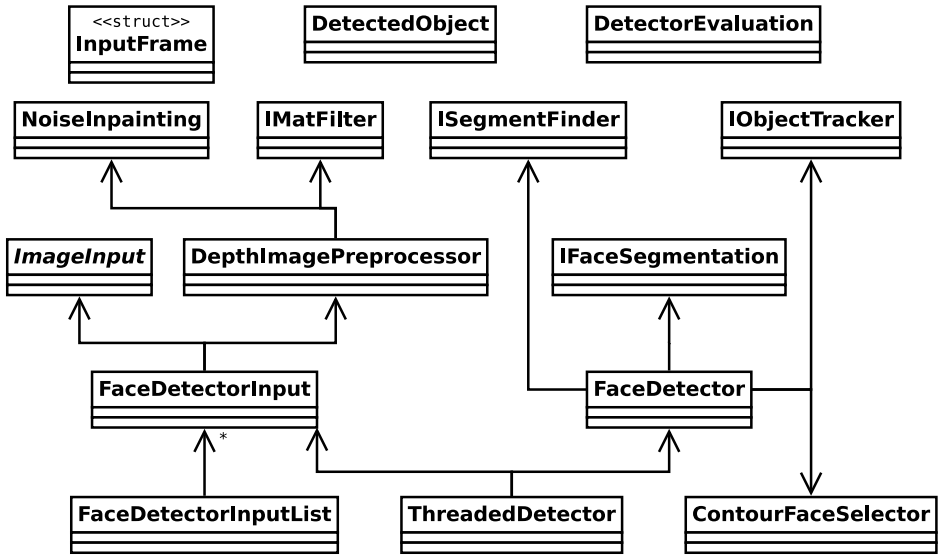
Všechny moduly, kromě samotné aplikace, jsou realizovány ve formě staticky linkované knihovny. Byly implementovány s ohledem na přenositelnost mezi platformami s výjimkou sledování paměťové náročnosti profilerem (popsáno dále).

3.2.1 Struktura kódu

Jak bylo nastíněno výše, majoritní část implementace detekce hlavy v hloubkových obrazech se nachází v modulu `DepthFaceDetectorLib`. Třídní diagram popisující strukturu a závislosti nejdůležitějších tříd je zachycen v diagramu na obrázku 20.

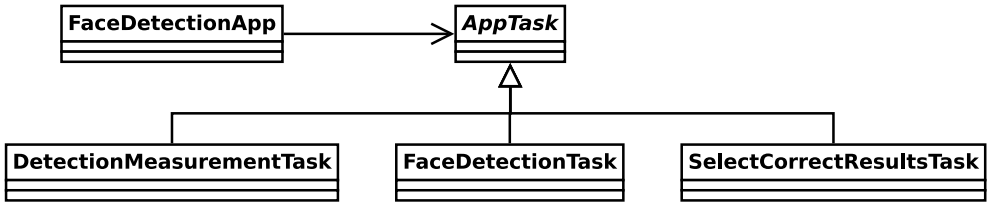
Těmito třídami a strukturami jsou:

- `InputFrame` - reprezentuje vstupní hloubkový snímek určený k segmentaci. Obsahuje originální obraz, jeho zpracovanou variantu (normalizace hodnot, atd.) a případný doprovodný světelný snímek. Tato struktura je používaná velkým množstvím ostatních funkcí tohoto modulu, z důvodu přehlednosti proto nejsou tyto vazby na obrázku 20 zachyceny.
- `DetectedObject` - reprezentuje nalezený, detekovaný objekt. Podobně, jako struktura reprezentující vstupní snímek, tato třída je používána větším množstvím funkcí a nejsou tyto vazby zobrazeny v třídním diagramu.
- `DetectorEvaluation` - vyhodnocuje výsledky konfigurace detektoru na základě poskytnutých výsledků detekce jednotlivých snímků, které mají (uživatelem) přiřazené očekávané výsledky.

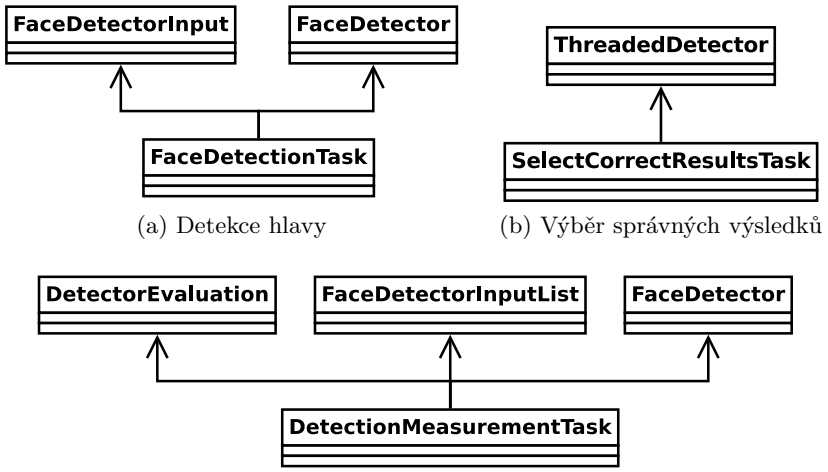


Obrázek 20: Třídní diagram - implementace detekce hlavy v hloubkových obrazech

- `NoiseInpainting` - retušuje šum v hloubkových obrazech - rekonstruuje malé oblasti pixelů s hodnotou h_u.
- `IMatFilter` - rozhraní pro algoritmy filtrující obrázek.
- `IObjectTracker` - rozhraní pro algoritmy vyhledávající detekovaný objekt mezi snímky.
- `ISegmentFinder` - rozhraní pro algoritmy vyhledávající objekty ve vstupním obraze bez dodatečných informací. Výstupem jsou markery.
- `IFaceSegmentation` - rozhraní pro algoritmy segmentace; vstupem je marker, výstupem je přesně segmentovaný objekt.
- `ContourFaceSelector` - z množiny nalezených objektů vybírá ten, který odpovídá lidské hlavě na základě vektoru chyb, jak bylo popsáno dříve.
- `ImageInput` - bázová abstraktní třída pro různé typy zdrojů vstupních obrázků.
- `DepthImagePreprocessor` - upravuje vstupní snímek tak, aby byl připravený pro další zpracování. Především jde o normalizaci hodnot pixelů a případné filtrování.
- `FaceDetectorInput` - zdroj vstupních snímků pro detekci hlavy; využívá zdroj obrázků, které připraví pro další zpracování. Volitelně umožňuje přeskočit nezajímavé snímky.
- `FaceDetectorInputList` - zdroj vstupních snímků, který je implementován jako větší množství jednotlivých zdrojů.
- `FaceDetector` - implementuje celý postup detekce hlavy od načtení, pokus o nalezení hlavy z předchozího snímku, přes detekci markerů, segmentaci a klasifikaci.



Obrázek 21: Třídní diagram - aplikace



(a) Detekce hlavy

(b) Výběr správných výsledků

(c) Měření kvality detekce

Obrázek 22: Třídní diagram - hlavní závislosti úloh aplikace na modulu detekce hlavy

- `ThreadedDetector` - vykonává paralelně detekci hlavy v několika snímcích, každý obraz je zpracováván v samostatném vlákně.

Dílčí kroky detekce jsou reprezentovány rozhraními, jejichž implementací je možno zavést konkrétní typ algoritmu. Díky tomu je možné snadno rozšířit aplikaci o další typy algoritmů vytvořením jedné třídy a tedy bez nutnosti opatrně upravovat kód na mnoha samostatných, vzájemně provázaných místech. Vytvoření instance konkrétního typu třídy je realizováno formou továren dle stejnojmenného návrhového vzoru. Algoritmy jsou identifikovány svým názvem, který uživatel uvede v konfiguračním souboru.

Diagram popisující strukturu implementace samotné aplikace detektoru hlavy je na obrázku 21. Úkoly, které program umí provést, jsou reprezentovány samostatnými třídami. Nejdůležitější vazby mezi těmito úlohami a modulem pro detekci jsou zachyceny v diagramu na obrázku 22.

- `AppTask` - abstraktní bázová třída implementací jednotlivých úloh.
- `FaceDetectionApp` - implementace aplikace detektoru hlavy. Načítá konfigurační soubor a spustí úlohu dle konfigurace. Poskytuje továrny implementací různých algoritmů.
- `FaceDetectionTask` - úloha detekce hlavy zobrazující detailní výsledky jednotlivých kroků algoritmu při aktuální konfiguraci.

- `SelectCorrectResultsTask` - úloha výběru správných výsledků detekce jednotlivých snímků.
- `DetectionMeasurementTask` - úloha měření kvality detekce.

3.3 Implementace vybraných oblastí

3.3.1 Profiler

Profiler měří časovou a paměťovou náročnost jednotlivých kroků implementace a jeho primární využití je v úkolu měření kvality detekce, jehož výstup je využívaný v následující kapitole - Výsledky.

Měření časové náročnosti je realizováno pomocí časovačů (třída `std::chrono::high_resolution_clock` v C++11) a výpočtu rozdílu času před a po vykonání dané funkce. Tato metoda není tak přesná, jako provedení cyklu opakovaného vykonání daného kódu a následné průměrování změřeného času, byla však zvolena z následujících důvodů:

- Nejrychlejší kroky trvají řádově v desítkách ms, což už je dost vysoká hodnota, aby bylo možno zanedbat zmíněnou nepřesnost.
- Některé algoritmy trvají déle (především One Cut) a opakování výpočtu by násobně zvyšovalo časovou náročnost měření.
- U implementace základního grafového řezu nastal jev, kdy opakovaný výpočet nad stejnými daty byl 7 až 10. rychlejší, než první iterace. Pravděpodobně se jedná o vliv mezipaměti procesoru a měření cyklu by sice lépe měřilo výkon samotné implementace, bylo by ale méně vypovídající o výsledcích v reálné situaci, kdy není potřeba zpracovávat opakovaně ta samá data stejným způsobem.

Měření paměťové náročnosti monitoruje alokace, realokace a dealokace. Tato funkcionalita není přenositelná mezi platformami, protože závisí na implementaci gcc; funkce malloc, realloc a free jsou tzv. slabě definované, což znamená, že jestliže programátor poskytne vlastní implementaci, budou se volat tyto funkce [35]. Tohoto profiler využívá ke sledování jednotlivých požadavků a zároveň je přeposílá systému prostřednictvím knihovny libc.

Při případném převodu aplikace na jinou platformu je potřeba tuto část implementovat pro daný systém, nebo ji v kódu vypnout a profiler nebude podávat informace o paměťovém využití.

3.3.2 Zdroje vstupních snímků

Aplikace implementuje následující typy zdrojů vstupních obrázků:

- Lokální video soubor

- Složka obrázků - zdroj je definovaný adresou lokálního adresáře a regulárním výrazem, který specifikuje množinu souborů, které se mají načíst a předat ke zpracování.

Všechny zdroje implementují podporu druhého, volitelného, zdroje světelných snímků, které odpovídají hloubkovému obrazu a který je použit pro vizualizaci výsledku detekce, který je pro lidské vnímání přirozenější v barevném obraze.

Protože základem zdroje vstupních snímků je `ImageInput`, abstraktní bázová třída, je možno jednoduše rozšířit aplikaci o další typy zdrojů, jako je např. kamera připojená přes USB rozhraní, načítání dat přes počítačovou síť, zdroj syntetických snímků aj.

Ke každému snímku jsou přiřazena dodatečná data, která popisují zda a kde se v obraze nachází lidská hlava. Zdrojem těchto dat je uživatelský vstup v úloze pro výběr správných výsledků, data jsou ukládána v YAML formátu ve složce, kde se nachází zdrojový obrázek.

3.3.3 Detektor objektů

Jsou implementovány tyto typy detektorů objektů:

- Kaskádový klasifikátor s příznaky Haar - tento algoritmus, jako jediný, vyžaduje, aby zdroj snímků poskytoval i světelný obraz a v něm vyhledává lidskou hlavu. Hlavní využití tohoto zdroje markerů je v úloze výběru správných výsledků detekce a po té k měření přesnosti detekce pro různé konfigurace.

- Detekce objektů pomocí distanční transformace - implementováno podle popisu v teoretické sekci.

3.3.4 Segmentace hlavy

Samotou segmentaci je možno provést následujícími algoritmy:

- Základní grafový řez - aplikace využívá knihovny maxflow [31], která je implementována dle [1].

- One Cut - taktéž implementován pomocí maxflow a příkladu na [31].

- Grab Cut - tento algoritmus je implementován v upravené podobě v knihovně OpenCV. Na rozdíl od základního popisu tohoto algoritmu nahrazuje klasifikaci pixelu u (neznámá) klasifikacemi u_f - možné popředí a u_b - možné pozadí.

  Rozdělení pixelů pro výpočet GMM je pak adekvátně upraveno; pixely s klasifikací u_f jsou přiřazeny do množiny popředí a u_b jako pozadí (místo původního zařazení všech pixelů u do popředí).

- Watershed - je taktéž implementován v OpenCV. V aplikaci je k dispozici ve dvou variantách:

– Jedna iterace - všechny vstupní markery jsou použity jako markery pro watershed a výsledek je interpretován jako segmentace všech objektů.
– Každý marker samostatně - vstupní maskou je vždy jen jeden marker, okolo kterého je vykreslen prstenec definující oblast okolo objektu, která je již považována za pozadí. Tento způsob se snaží řešit problém, kdy Watershed používá celý snímek pro segmentaci a vyžaduje tak označení všech oblastí okolo hlavy.

4 Výsledky

Většinu kroků a parametrů detekce je možno v aplikaci nastavit pomocí konfiguračního souboru, včetně výběru konkrétních algoritmů. V průběhu vývoje bylo vytvořeno několik videí určených k ověření jednotlivých postupů a různých nastavení.

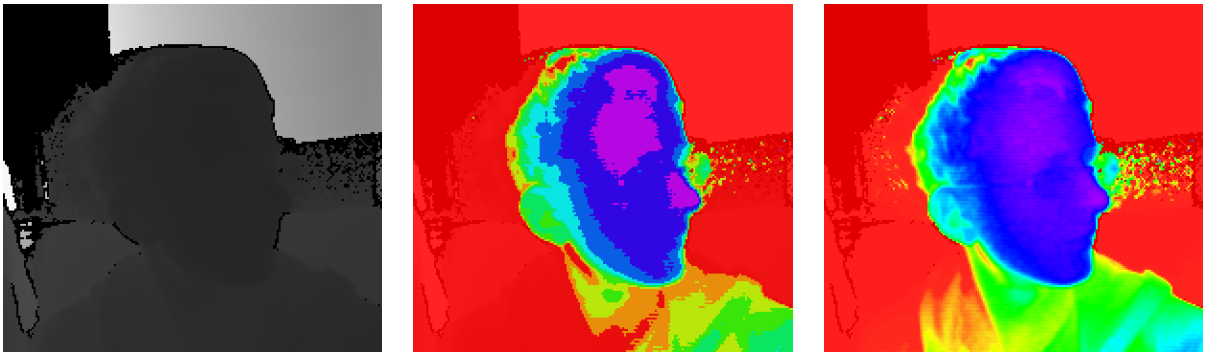
4.1 Měření

Měření bylo provedeno pomocí implementované aplikace. Nejprve byly zadány a uloženy očekávané výsledky ve formě oblastí - kontur. Poté byla spuštěna úloha měření detekce, která automaticky postupně detekovala jednotlivé snímky z připravených videí, měřila jejich časovou a paměťovou náročnost a úspěšnost detekce. Tato úloha byla spuštěna opakovaně pro různé konfigurace. Výstupem měření konfigurace jsou následující hodnoty:

- Přesnost - poměr správných výsledků; relativní množství snímků, u nichž byla detekce úspěšná. Za úspěšnou se považuje, když byla hlava detekována ve správné oblasti a byla očekávána, nebo pokud nebyla detekována a žádná se ve snímku nenachází. Některé snímky jsou sporné, např. hlava je částečně zakrytá; obrázky takto označené se neberou v potaz.
- Průměrná podobnost se vzorem - v případě, že je ve snímku očekávána hlava, je porovnána detekovaná oblast s oblastí očekávanou. Podobnost je vyjádřena následujícími typy hodnot:
  - Podobnost (správné) - je měřena jen pro správné výsledky, tedy v případě, že byla hlava nalezena a zároveň očekávána.
  - Podobnost (celkově) - je počítána pro všechny snímky, kdy byla hlava očekávána. Jestliže nebyla nalezena, je podobnost pro tento případ definována jako 0.0.
- Čas zpracování (ms) - průměrná doba v milisekundách, po kterou se jeden snímek zpracoval, od načtení ze souboru, přes všechny kroky detekce a segmentace.
- Čas markeru (ms) - průměrná doba v milisekundách, po kterou se segmentoval jeden marker.
- Paměť (MB) - průměrná alokace paměti v průběhu zpracování jednoho snímku v megabytech.
- Počet markerů - průměrný počet markerů, který byl nalezen v jednom snímku. Toto číslo závisí na jednotlivých snímcích, jejich složitosti.

Použitá sestava:

- CPU - Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz
- RAM - 8GB, 1600MHz



(a) Nezpracovaný snímek

(b) Barevně zvýrazněné úrovně hloubky 8bitového snímku

(c) Barevně zvýrazněné úrovně hloubky 16bitového snímku

Obrázek 23: Ukázka rozdílu v rozlišení barevné hloubky 8 a 16bitového obrazu

4.2 Výsledky kroků detekce hlavy

Jednotlivé kroky detekce hlavy byly podrobeny měření kvality a výkonu s různými konfiguracemi daného úkonu pro porovnání metod. Kromě zhodnocení výsledků jsou popsány praktické zkušenosti a problémy, které se projevily v průběhu vývoje a experimentování.

4.2.1 Vstupní obraz

Při práci s obrazy v počítačích je nejčastěji používaná 8bitová hloubka hodnot složek pixelů, v případě hloubkových obrazů to znamená 255 různých hodnot vzdálenosti[9]. Tento rozsah se prakticky ukázal být příliš malý při segmentaci lidské hlavy, protože je většinou reprezentována jen několika málo hodnotami, jak je znázorněno na příkladu na obrázku 23b.

Na příkladu je možno si všimnout, že na některých místech nelze hranu detekovat rozdílem hodnot pixelů, protože ten je nanejvýš 1, ale hustotou změny hodnoty na větší ploše. Tento problém lze částečně kompenzovat Gaussovským rozostřením, po jehož aplikování hodnota pixelu je ovlivněna svým okolím. Tento zisk se ovšem kompletně ztratí, pokud by výsledný snímek byl v paměti reprezentován 8bitovými hodnotami, proto byl zvolen desetinný typ (`float`) a normalizace hodnot do rozsahu 0.0 - 1.0, jak bylo popsáno dříve.

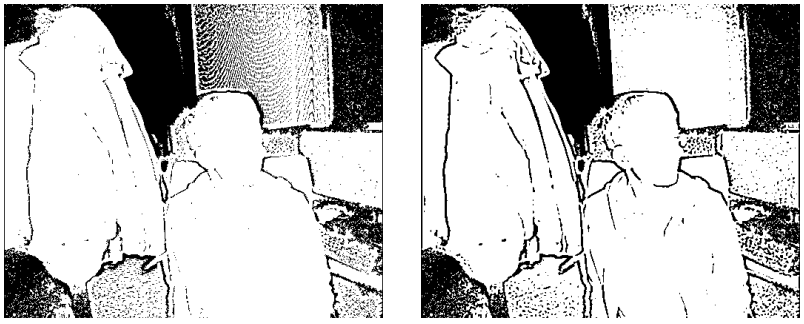
Výstupem mnohých zařízení je obraz o 16bitové hloubce hodnot pixelů, která poskytuje výrazně jemnější rozlišení vzdáleností - teoreticky až 65 535 různých hodnot. Prakticky tato zařízení nejsou tak citlivá a nevyužívají tedy plně všech hodnot v tomto rozsahu, ale přesto je jejich rozlišení hloubky obvykle vyšší a kompenzace filtrováním není potřeba, nebo alespoň ne v takové míře. Příklad rozdílu v barevné hloubce snímků pro účely detekce hlavy je na obrázku 23.

[9] Jedna hodnota je obvykle přiřazena významu „neznámá hloubka“.

Oddělení objektů	Přesnost	Podobnost (správné)	Podobnost (celkově)
Laplacián	0.83	0.78	0.7
Adaptivní prahování	0.8	0.76	0.68

Oddělení objektů	Čas zpracování ms	Čas markeru ms	Paměť MB	Počet markerů
Laplacián	651	39	4.9	16.2
Adaptivní prahování	580	41	4.6	13.8

Tabulka 1: Měření oddělení jednotlivých objektů



(a) Adaptivní prahování (b) Laplacián

Obrázek 24: Příklad oddělení objektů

4.2.2 Kaskádový klasifikátor Haar

Jestliže je k dispozici světelný obraz, jedná se o velmi rychlou metodu, jak nalézt hlavu ve snímku a pak ji dále přesněji segmentovat. Spolehlivost závisí na charakteristice světelných snímků - jestliže je barevná kamera fyzicky příliš vzdálena hloubkovému zařízení, je nutno provést korekci obrazu, která ovšem stále nese zkreslení v závislosti na vzdálenosti objektu.

4.2.3 Detekce objektů pomocí distanční transformace

Prvním krokem je oddělení jednotlivých objektů, které může být provedena laplaciánem nebo adaptivním prahováním. Porovnání těchto přístupů je v tabulce 1 a příklad výstupu na obrázku 24. Ačkoliv je výsledná kvalita detekce při optimálních parametrech podobná a adaptivní prahování je lehce rychlejší, experimentování s různými parametry ukázalo, že kvalita detekce hlavy s použitím tohoto algoritmu je výrazně více nestabilní při menších změnách parametrů, zatímco laplacián je robustnější.

Filtry aplikované na obrázek s detekovanými hranami před distanční transformací jsou porovnány v tabulce 2; Pro filtrování obrazu před distanční transformací se ukázalo být vhodné Gaussovské rozostření; snížení šumu nelokální střední hodnotou dává sice velmi podobné vý-

Filtr	Přesnost	Podobnost (správné)	Podobnost (celkově)
Gauss	0.83	0.78	0.7
Mean	0.82	0.78	0.7
Medián	0.73	0.68	0.6
Žádný	0.8	0.69	0.62

Filtr	Čas zpracování ms	Čas markeru ms	Paměť MB	Počet markerů
Gauss	651	39	4.9	16.2
Mean	1,251	41	396	14
Medián	767	51	5	14.7
Žádný	460	35	4.4	12.7

Tabulka 2: Měření různých filtrů před distanční transformací

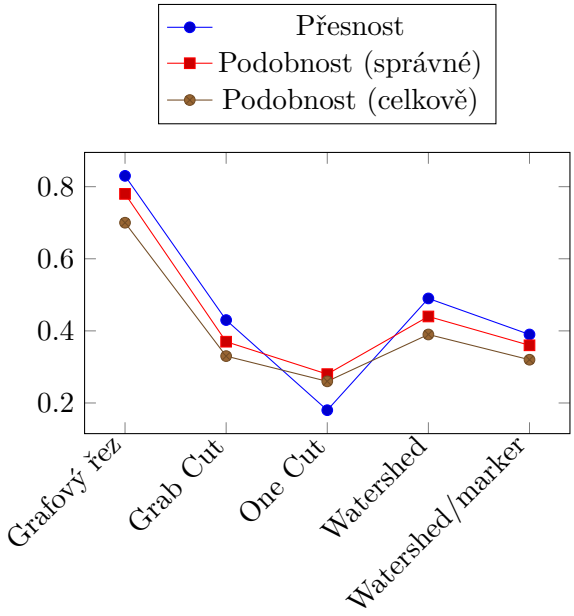
Hledání objektů v distanční transformaci	Přesnost	Podobnost (správné)	Podobnost (celkově)
Analýza	0.83	0.78	0.7
Prahování	0.57	0.49	0.44

Hledání objektů v distanční transformaci	Čas zpracování ms	Čas markeru ms	Paměť MB	Počet markerů
Analýza	651	39	4.9	16.2
Prahování	616	77	4.5	7.6

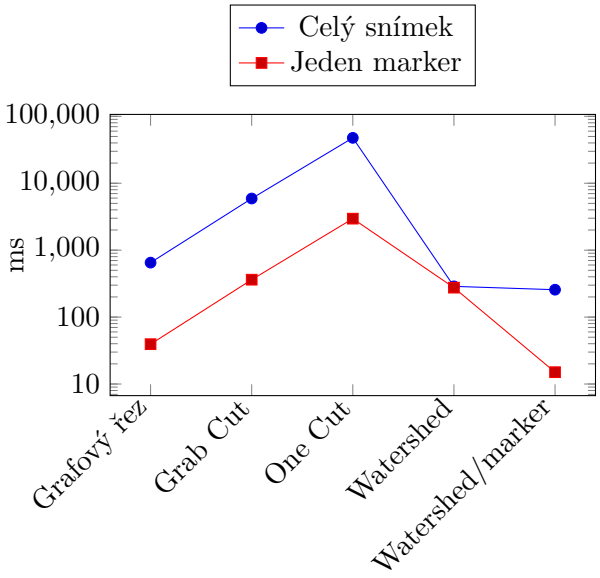
Tabulka 3: Měření různých způsobů hledání objektů ve výsledku distanční transformace

sledky, je ale výrazně pomalejší a to přesto, že implementace tohoto algoritmu v OpenCV silně využívá paralelizaci. Nejrychlejší detekce je, celkem pochopitelně, když není aplikován žádný filtr. Úspěšnost nalezení hlavy je sice velmi podobná, jako v ostatních případech, přesnost ale velmi klesá, což je způsobeno šumem, který se použitím filtrů snažíme odstranit.

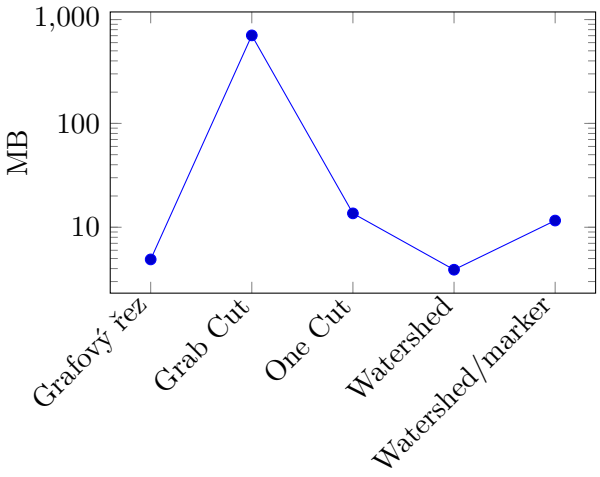
Posledním krokem je nalezení jednotlivých objektů ve výsledku distanční transformace; jednotlivé implementované způsoby jsou porovnány v tabulce 3. Použití prahování dává dobré výsledky pouze v případě snímků, kdy je výrazná hrana okolo celé hlavy, což často není splněno v oblasti mezi hlavou a trupem a tedy celková úspěšnost i přesnost jsou nízké. Právě z tohoto důvodu byla navržena hlubší analýza vrcholů distanční transformace, která tento problém řeší úspěšněji. Praktickým problémem je ale větší množství nalezených markerů, které jsou dále segmentovány, čímž tato metoda nepřímo výrazně zvětšuje výpočetní náročnost celého zpracování jednoho snímku.



(a) Přesnost detekce



(b) Časová náročnost (log. měřítko)



(c) Paměťová náročnost (log. měřítko)

Obrázek 25: Měření různých typů segmentace hlavy

	Grafový řez	Grab Cut	One Cut	Watershed	Watershed/marker
Přesnost	0.83	0.43	0.18	0.49	0.39
Podobnost (správné)	0.78	0.37	0.28	0.44	0.36
Podobnost (celkově)	0.7	0.33	0.26	0.39	0.32
Čas zpracování ms	651	5,911	47,638	288	256
Čas markeru ms	39	362	2,956	275	15
Paměť MB	4.9	704.2	13.6	3.9	11.6
Počet markerů	16.2	16.2	16.1	16.2	16.2

Tabulka 4: Měření různých typů segmentace hlavy

4.2.4 Segmentace obrazu

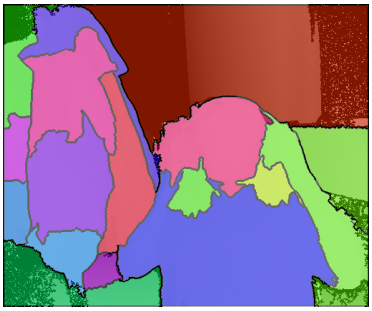
Jednotlivé algoritmy segmentace obrazu jsou porovnány v tabulce 4 a ve grafu 25 (pro paměťovou a časovou náročnost bylo použito logaritmické měřítko).

Měření bylo provedeno s následující konfigurací detekce objektů, která byla nalezena jako nejúspěšnější:

- Oddělení objektů: Laplacián
- Filtr před distanční transformací: Gausián.
- Hledání objektů v distanční transformaci: analýza vrcholů d.t.

Nejlepší výsledky podává grafový řez v základní variantě. Při definici váhy hran mezi vrcholy pixelů v grafu je použita funkce $B_{p,q}(L_p, L_q)$, která odpovídá vážené Gaussovské funkci rozdílu hodnot sousedících pixelů. V původním návrhu grafového řezu je rozptyl pro tuto funkci definován jako průměrný rozptyl v obraze, prakticky ale tento parametr gausiánu nedává stabilní výsledky pro účely segmentace hlavy v hloubkovém snímku. Protože z hlediska hloubkových obrazů je model hlavy dostatečně specifický, nelze např. předpokládat prudkou změnu v hodnotě hloubky, lze rozptyl definovat jako parametr modelu hlavy. Empiricky byla nalezena hodnota $\sigma^2 = 1.845 \cdot 10^{-6}$ (jedná se o bezrozměrnou hodnotu vzhledem k tomu, že hodnota pixelů v obraze je po normalizaci bezrozměrná a odpovídá 0.12 při rozsahu hodnot pixelů $0 - 255$).

One Cut přidáním vrcholů binů zavádí do systému podmínku hledání globálního optima, které dobře funguje u světelných snímků, kdy se předpokládá, že objekt je složen z podobných barev, které z větší míry nesdílí s pozadím. Tento předpoklad ale u hloubkových snímků neplatí; situace, že se ve scéně nachází oblast v podobné hloubce, jako člověk, je pravděpodobná a naru-



(a) Všechny markery jsou použity pro jeden běh

(b) Každý marker je zpracován samostatně

Obrázek 26: Příklad nepřesnosti Watershed

Hledání objektů mezi snímky	Přesnost	Podobnost (správné)	Podobnost (celkově)	Plné detekce
Žádné	0.83	0.78	0.7	1
Význačné body	0.85	0.79	0.7	0.58
Šablona	0.86	0.78	0.7	0.2

Hledání objektů mezi snímky	Čas zpracování ms	Čas markeru ms	Paměť MB	Počet markerů
Žádné	651	39	4.9	16.2
Význačné body	504	40	8.3	16.7
Šablona	175	46	7.1	12

Tabulka 5: Měření různých způsobů sledování objektů mezi snímky

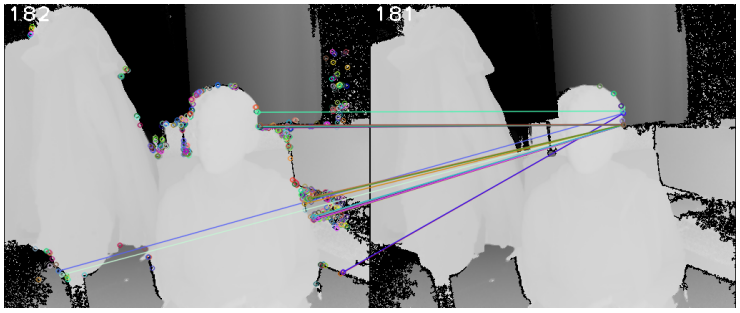
šuje segmentaci. Přidané biny navíc velmi výrazně zvětšují výpočetní náročnost a při segmentaci většího množství markerů se algoritmus stává příliš pomalým.

Z důvodu hledání globálního optima i Grab Cut, který je navržený pro RGB snímky, nedává dobré výsledky; je-li ve snímku přítomen objekt v podobné hloubce jako segmentovaná hlava, mohou nalezené parametry modelu směsi Gaussovských funkcí vést ke klasifikaci hlavy jako pozadí.

Obě varianty Watershed jsou velmi rychlé, jejich úspěšnost je ale horší. To je způsobeno tím, že tento algoritmus segmentuje obraz tak, že využije všechny pixely snímku a tedy pro dobré výsledky je potřeba poskytnout markery pro všechny oblasti okolo hlavy. Obrázek 26 demonstruje tento problém.

4.2.5 Sledování objektů mezi snímky

Tabulka 5 porovnává vliv různých způsobů sledování objektů mezi snímky na detekci hlavy. Parametr „plné detekce“ vyjadřuje relativní množství snímků, u kterých byla provedena kompletní detekce hlavy, což se děje v případech, že algoritmus hlavu na základě předchozího snímku



Obrázek 27: Ukázka nedostatečného počtu význačných bodů a párování pro sledování hlavy mezi snímky. Napravo je původní obrázek s připravenými body, vlevo nový snímek.

nenašel, když se jedná o první obrázek vstupu, nebo pokud v předchozím snímku hlava nebyla nalezena.

Při použití šablony došlo k výraznému zrychlení zpracování obrazu a překvapivě zároveň k lehkému zvýšení přesnosti. To je způsobeno tím, že v mnohých snímcích, kde by standardní postup selhal, byla nalezena hlava pomocí vyhledání na základě výstupu předchozí detekce.

Použití význačných bodů výrazně detekci neovlivňuje, protože velmi často selhává. Důvodem je odlišná charakteristika hloubkových obrazů oproti snímkům světelným, pro které tato metoda byla vyvinuta; obsahují méně výrazné body a párování dává špatné výsledky kvůli větší podobnosti nesouvisejících oblastí. Příklad tohoto problému je na obrázku 27.

5 Závěr

Byla navržena a implementována aplikace k plně automatizované detekci hlavy v hloubkových obrazech s využitím grafových řezů, která je schopna aplikovat různé algoritmy jednotlivých kroků detekce a nastavovat jejich parametry dle konfigurace. Měření na testovací množině snímků ukázalo, že nejúspěšnější je základní varianta grafového řezu s průměrnou přesností 80%. Specializované algoritmy Grab Cut a One Cut jsou optimalizované pro světelné obrazy a na hloubkových datech nedávají tak dobré výsledky. Taktéž byl pro srovnání testován algoritmus Watershed, který sice segmentuje obraz zdaleka nejrychleji, jeho přesnost je ale velmi závislá na detekování všech oblastí okolo segmentovaného objektu a v implementované aplikaci kvůli tomu měl výrazně horší výsledky.

Implementace má stále potenciál pro zlepšení kvality detekce, aktuálně má např. horší výsledky, když má člověk na hlavě větší doplňky. Možnými úpravami jsou např.:

- Využití paralelizace - aktuálně aplikace zpracovává více snímků zároveň jen v úloze zadání očekávaných výsledků. Protože jsou jednotlivé kroky od sebe izolované, je možno implementaci upravit tak, aby po dokončení výpočtu jednoho kroku se zahájil tento krok pro další snímek, zatímco výsledky byly paralelně zpracovány dalším krokem.

  Taktéž je možno paralelizovat zpracování jednotlivých markerů.

- Lepší klasifikace - model hlavy pro účely klasifikace je velmi zjednodušený a je zde prostor pro vyhledávání specifických příznaků.

  V případě dostupnosti dostatečně velké a variabilní množiny trénovacích vzorků by bylo možno zkusit zavést klasifikátor založený např. na umělých neuronových sítích nebo natrénovat kaskádový klasifikátor Haar na hloubkových snímcích.

- Optimalizace implementace
  - Využití specializovaných instrukcí na cílové platformě
  - Implementace v Open CL [36] nebo Cudě [37]
  - Paralelizace pomocí Open MP [38]

- Zúžený výběr markerů - výstupem kroku pro detekci objektů je množina markerů, každý z nich je pak samostatně segmentován. To zvyšuje výpočetní náročnost a další optimalizací tedy může být specializovaný výběr těch markerů, které mají větší pravděpodobnost, že jsou hledaným objektem, například analýzou rozptylu hodnot hloubky v dané oblasti.

Architektura aplikace byla zvolena tak, že vstupní zdroj snímků a algoritmy jednotlivých kroků detekce jsou realizovány implementací rozhraní, je tedy možno program snadno rozšířit o nové algoritmy a zdroje vstupních obrazů.

Literatura

[1] BOYKOV, Yuri a Vladimir KOLMOGOROV. *An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision.* IEEE Transactions on PAMI, Svazek 26, Číslo 9, 1124-1137, 2004. Dostupné také z: http://www.csd.uwo.ca/faculty/yuri/Papers/pami04.pdf

[2] BERTALMIO, Marcelo, Guillermo SAPIRO, Vicent CASELLES a Coloma BALLESTER. *Image Inpainting.* Proceedings of SIGGRAPH 2000. Dostupné také z: http://www.dtic.upf.edu/~mbertalmio/bertalmi.pdf

[3] M. BERTALMIO, A. L. BERTOZZI a G. SAPIRO. *Navier-Stokes, Fluid Dynamics, and Image and Video Inpainting.* IEEE CVPR 2001. Hawaii, USA, 2001. Dostupné také z: http://www.dtic.upf.edu/~mbertalmio/final-cvpr.pdf

[4] TELEA Alexandru, *An Image Inpainting Technique Based on the Fast Marching Method.* Journal of Graphics, GPU, and Game Tools, Svazek 9, Číslo 1, 23-34, 2004. Dostupné také z: https://pdfs.semanticscholar.org/622d/5f432e515da69f8f220fb92b17c8426d0427.pdf

[5] Smoothing Images. *OpenCV 2.4.13.6 documentation* [online]. 2018 [cit. 2018-03-03]. Dostupné z: https://docs.opencv.org/2.4/doc/tutorials/imgproc/gausian_median_blur_bilateral_filter/gausian_median_blur_bilateral_filter.html

[6] SZELISKI, Richard. *Computer Vision: Algorithms and Applications.* London: Springer, 2010. Texts in computer science. ISBN 978-1-84882-934-3.

[7] BUADES, Antoni, Bartomeu COLL a Jean-Michel MOREL. *Non-Local Means Denoising.* Image Processing On Line [online]. 208-212, 2011 [cit. 2018-03-03]. Dostupné z: http://www.ipol.im/pub/art/2011/bcm_nlm/

[8] VIOLA, Paul a Michael JONES. *Rapid object detection using a boosted cascade of simple features.* IEEE CVPR, 2001. Dostupné z: https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf

[9] LIENHART, Rainer. haarcascade_frontalface_alt.xml. *OpenCV - GitHub* [online]. San Francisco, 2005 [cit. 2017-04-06]. Dostupné z: https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_frontalface_alt.xml

[10] Image Segmentation with Distance Transform and Watershed Algorithm *OpenCV library* [online]. [cit. 2018-03-04]. Dostupné z: https://docs.opencv.org/3.3.0/d2/dbd/tutorial_distance_transform.html

[11] Image Segmentation with Watershed Algorithm. *OpenCV library* [online]. [cit. 2017-04-08]. Dostupné z: http://docs.opencv.org/3.1.0/d3/db4/tutorial_py_watershed.html

[12] Laplace Operator. *OpenCV 2.4.13.6 documentation* [online]. 2018 [cit. 2018-03-04]. Dostupné z: https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/laplace_operator/laplace_operator.html

[13] Image Thresholding. *OpenCV library* [online]. [cit. 2017-04-08]. Dostupné z: http://docs.opencv.org/3.1.0/d7/d4d/tutorial_py_thresholding.html

[14] D. M. GREIG, B. T. PORTEOUS a A. H. SEHEULT. *Exact maximum a posteriori estimation for binary images.* Journal of the Royal Statistical Society, Series B, Svazek 51, Číslo 2, 271–279, 1989.

[15] BOYKOV, Yuri a Marie-Pierre JOLLY. Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D Images. *International Conference on Computer Vision.* Svazek 1, 105-112, 2001. Dostupné také z: http://www.csd.uwo.ca/~yuri/Papers/iccv01.pdf

[16] ROTHER, Carsten, Vladimir KOLMOGOROV a Andrew BLAKE. *"GrabCut" — Interactive Foreground Extraction using Iterated Graph Cuts.* ACM Transactions on Graphics, Svazek 23, Číslo 3, 309-3014, 2004. Dostupné z: https://cvg.ethz.ch/teaching/cvl/2012/grabcut-siggraph04.pdf

[17] TANG, Meng, Lena GORELICK, Olga VEKSLER a Yuri BOYKOV. *GrabCut in One Cut.* International Conference on Computer Vision (ICCV), Sydney, Australia, 2013. Dostupné z: http://www.csd.uwo.ca/~ygorelic/iccv13_one_cut.pdf

[18] MEYER, Fernand. *Color Image Segmentation*, ICIP92, 1992.

[19] The Watershed Transformation. *Centre for Mathematical Morphology* [online]. Fontainebleau, 2010 [cit. 2017-04-08]. Dostupné z: http://cmm.ensmp.fr/~beucher/wtshed.html

[20] FITZGIBBON, Andrew a Robert FISHER. A Buyer's Guide to Conic Fitting. *Proceedings of the 5th British conference on Machine vision,* 513-522, 1995. Také dostupné z: https://pdfs.semanticscholar.org/19d9/becce00a500bdd1cad5a19f9f16175347096.pdf

[21] shapedescr.cpp. *GitHub* [online]. San Francisco, c2000 [cit. 2018-04-21]. Dostupné z: https://github.com/opencv/opencv/blob/master/modules/imgproc/src/shapedescr.cpp

[22] Human head. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-03-18]. Dostupné z: https://en.wikipedia.org/wiki/Human_head

[23] Template Matching. *OpenCV 2.4.13.6 documentation* [online]. 2018 [cit. 2018-03-24]. Dostupné z: https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template_matching/template_matching.html

[24] Feature Detection and Description. *OpenCV library* [online]. [cit. 2018-04-16]. Dostupné z: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_table_of_contents_feature2d/py_table_of_contents_feature2d.html

[25] Harris Corner Detection. *OpenCV library* [online]. [cit. 2018-04-16]. Dostupné z: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_features_harris/py_features_harris.html

[26] Harris Corner Detector. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-04-16]. Dostupné z: https://en.wikipedia.org/wiki/Harris_Corner_Detector

[27] CALONDER, Michael, Vincent LEPETIT, Christoph STRECHA a Pascal FUA. BRIEF: Binary Robust Independent Elementary Features. *In European Conference on Computer Vision*. 2010. Také dostupné z: https://www.labri.fr/perso/vlepetit/pubs/calonder_eccv10.pdf

[28] RUBLEE, Ethan, Vincent RABAUD, Kurt KONOLIGE a Gary BRADSKI. ORB: an efficient alternative to SIFT or SURF. ICCV 2011, 2564-2571. Také dostupné z: http://www.willowgarage.com/sites/default/files/orb_final.pdf

[29] MUJA, Marius a David G. LOWE. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. *International Conference on Computer Vision Theory and Application (VISSAPP'09).* 2009. Také dostupné z: https://www.cs.ubc.ca/research/flann/uploads/FLANN/flann_visapp09.pdf

[30] Random sample consensus. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-04-16]. Dostupné z: https://en.wikipedia.org/wiki/Random_sample_consensus

[31] Code. *Computer Vision at Western* [online]. London, Canada, 2016 [cit. 2017-04-08]. Dostupné z: http://vision.csd.uwo.ca/code/

[32] Simple .INI file parser in C, good for embedded systems. *GitHub* [online]. [2009] [cit. 2017-04-09]. Dostupné z: https://github.com/benhoyt/inih

[33] A YAML parser and emitter in C++. *GitHub* [online]. [2008] [cit. 2017-04-09]. Dostupné z: https://github.com/jbeder/yaml-cpp

[34] POSIX Threads. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, [2017] [cit. 2017-04-09]. Dostupné z: `https://en.wikipedia.org/wiki/POSIX_Threads`

[35] An alternative for the deprecated ___malloc_hook functionality of glibc. *Stack Overflow* [online]. 2013 [cit. 2017-04-09]. Dostupné z: `http://stackoverflow.com/a/17850402`

[36] OpenCL. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-04-22]. Dostupné z: `https://en.wikipedia.org/wiki/OpenCL`

[37] CUDA. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-04-22]. Dostupné z: `https://en.wikipedia.org/wiki/CUDA`

[38] OpenMP. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-04-22]. Dostupné z: `https://en.wikipedia.org/wiki/OpenMP`

A Souborová příloha

K této práci je přiloženo CD s aplikací a jejími zdrojovými kódy;

- `/app/`

  Adresář s aplikací, optimální konfigurací a příkladem snímků

- `/app/video/photoInput/`

  Adresář s hloubkovými obrazy

- `/app/DepthFaceDetector`

  Výsledný program

- `/app/config.ini`

  Konfigurace detekce; jednotlivé položky jsou komentovány

- `/app/photoInput.ini`

  Konfigurace vstupu pro načítání obrázků ze složky `/app/video/photoInput/`

- `/app/GuiConfig.txt`

  Konfigurace GUI - vypíná zobrazení některých typů kroků pro větší přehlednost

- `/app/readme.txt`

  Základní návod k používání aplikace s příklady

- `/app/haarcascade_eye_tree_eyeglasses.xml`
  `/app/haarcascade_frontalface_alt.xml`

  Soubory kaskádového klasifikátoru Haar

- `/app/DescriptorExtractorParams.yml`
  `/app/FeatureDetectorParams.yml`

  Konfigurace detekce a popisu význačných bodů pro jednotlivé typy algoritmů, které s význačnými body pracují

- `/src/`

  Adresář se zdrojovými kódy aplikace

- `/src/Core/`
  `/src/DepthFaceDetector/`
  `/src/DepthFaceDetectorLib/`
  `/src/INI/`

`/src/OpenCVUtility/`
`/src/Profiler/`
`/src/YamlUtility/`

Adresáře se zdrojovými kódy jednotlivých modulů; každý modul obsahuje soubory projektu programu Eclipse

- `/src/makefile`

  Soubor pro automatickou kompilaci s použitím programu `make`

- `/src/readme.txt`

  Návod ke kompilaci aplikace a seznam závislostí